NAT'L INST. OF STAND & TECH
A11106 402802
NIST PUBLICATIONS
REFERENCE

FILE COPY
NOT REMOVE
MAR 21 198

Dir. 731

NBSIR 80-1978 (R)

Initial Graphics Exchange Specification IGES Version 1.0

Roger N. Nagel, Ph.D., Project Manager
Walt W. Braithwaite, M.S., Boeing
Philip R. Kennicott, Ph.D., General Electric

Programmable Automation Group
Mechanical Processes Division
Center for Mechanical Engineering and Process Technology
National Engineering Laboratory
National Bureau of Standards

March 1980

DEPARTMENT OF COMMERCE
ONAL BUREAU OF STANDARDS

QC
100
.U56
NO.80-1978
(R)
1980

ACKNOWLEDGEMENT

This report contains an Initial Graphics Exchange Specification (IGES). The National Bureau of Standards under contract to the Air Force, Army, Navy, and NASA is coordinating and directing the effort on IGES. Walt W. Braithwaite of Boeing, Philip R. Kennicott of General Electric and Roger N. Nagel of the National Bureau of Standards (NBS) served as the technical committee in formulating IGES. Both The Boeing Company and The General Electric Company have contributed heavily to the IGES effort and have not received compensation from the NBS or its funding agencies. The contributions from Boeing and General Electric have been in the form of ideas, documents and travel, as well as extensive time spent on IGES by Walt Braithwaite, Philip Kennicott and their respective staffs. Without this essential support this IGES report would not have been possible.

The technical committee wishes to acknowledge and thank Marty Auman (NBS), John Lewis (GE), and David Moore (Boeing) for their valuable contributions to the IGES effort.

NBSIR 80-1978 (R)

INITIAL GRAPHICS EXCHANGE SPECIFICATION IGES VERSION 1.0

Roger N. Nagel, Ph.D., Project Manager
Walt W. Braithwaite, M.S., Boeing
Philip R. Kennicott, Ph.D., General Electric

Programmable Automation Group
Mechanical Processes Division
Center for Mechanical Engineering and Process Technology
National Engineering Laboratory
National Bureau of Standards

March 1980

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick, *Secretary*

Luther H. Hodges, Jr., *Deputy Secretary*
Jordan J. Baruch, *Assistant Secretary for Productivity, Technology, and Innovation*

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

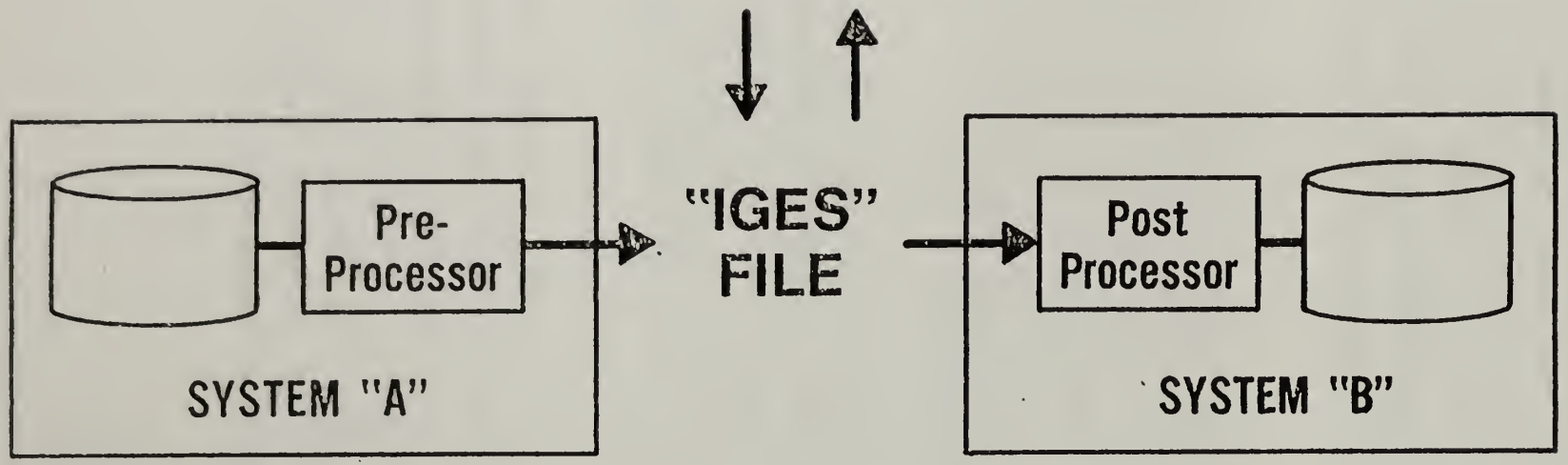
1.0 INTRODUCTION

This Initial Graphics Exchange Specification (IGES) report is the result of a three month intensive effort by the technical committee and many others who have worked with the committee and provided input and feedback. The IGES by design is an initial attempt at providing a specification for the exchange of data between Computer Aided Design/Computer Aided Manufacturing (CAD/CAM) systems. Figure 1 shows a functional description of the IGES concept.

The idea to create an IGES as an immediate step in the solution of the CAD/CAM data exchange problem was developed at the DOD MTAG meeting in Detroit on September 12-14, 1979. The CAD/CAM workshop at that meeting recommended that a meeting be held within thirty days to formulate an IGES early in 1980. As a result of that recommendation, a meeting was convened by the Air Force, Army, Navy, NASA and NBS at the National Academy of Sciences on October 11, 1979.

The October 11 meeting was used as a forum for the discussion of the IGES concept. Presentations were made by vendors, corporate systems designers, and standards groups on their respective efforts to solve the data exchange problem between CAD/CAM systems. At the conclusion of the one day meeting, it was generally agreed that an initial graphics exchange specification was needed immediately.

Roger Nagel of the NBS was asked to chair an effort to produce an IGES early in 1980. Walt Braithwaite of Boeing and Philip Kennicott of General Electric were asked to serve as members of a technical committee, and the Boeing and GE corporate CAD/CAM exchange systems were selected as initial systems on which to build IGES. The Air Force, Army, Navy, and NASA have funded the NBS to direct and coordinate the IGES effort. GE and Boeing volunteered to pay their own individual expenses in the formulation of IGES.



FUNCTIONAL IMPLEMENTATION OF IGES

FIGURE 1

Subsequent to the October 11th meeting the technical committee conducted an intensive effort to refine the initial document into an acceptable IGES. Two public forums were held to gather comment and additional input. The first meeting on November 20 was held at GE for companies who would eventually write IGES translators, and the second meeting was held on December 14 at NBS for anyone interested in the IGES concept.

As a result of these meetings, the comments generated, and the work of the technical committee, IGES was formulated. It is important to note that IGES was created to meet an immediate need. IGES is a specification and not a standard. Efforts are underway to coordinate the IGES with standards making bodies. In particular an IGES working group is being organized to (among other tasks) coordinate with the various American National Standards Institute (ANSI) committees and other standards efforts. It is expected that the IGES will be one of the many important steps in the complex procedure required to produce a national standard.

In the short time span from the start of the IGES effort, several misconceptions about IGES have developed. In order to clarify them, the following lists of what the IGES is and is not are presented.

IGES is a specification for the exchange of data between CAD/CAM systems.

IGES is a vehicle for archiving data from a CAD/CAM system.

IGES is designed with the technical aspects of several current CAD/CAM systems in mind. Thus the translation from vendor systems to IGES and vice versa will not be one for one, but should be feasible.

IGES is based on the Boeing CAD/CAM Integrated Information Network, the General Electric Neutral Data Base, and a variety of other data exchange formats which were given to the committee.

IGES is the best specification that could be produced in the time frame permitted. While it is not a copy of any of the exchange formats presented to the committee, it has the advantage of the experience and knowledge gained in their production and use.

IGES is a set of geometrical, drafting, structural and other entities. Thus it has the capability to represent a majority of the information in CAD/CAM systems.

IGES is extensible. Several definition mechanisms have been provided to permit IGES to be expandable. A working committee has been set up to coordinate expansions and to correct errors.

IGES is not a national standard.

IGES is not a data base structure.

IGES is not designed for the technical aspects of any one of the currently available CAD/CAM systems.

IGES is not perfect, or the solution to all data exchange problems between CAD/CAM systems.

IGES is not a carbon copy of any of the exchange formats given to the technical committee.

IGES is not a stand alone contractual specification for a deliverable product.

IGES is not a complete specification of all the data in all CAD/CAM systems. Thus there may be a loss of data or structural information in the translation to and from IGES.

In summary, IGES is not perfect, it will not solve all the information exchange needs of CAD/CAM systems, and it will need future extension

beyond its current definition. However IGES goes a long way toward alleviating the current data exchange problem, and is a significant response to today's needs.

1.1 IGES Overview

This report is a reference manual for IGES. In the remainder of this chapter, the goals used in the definition of IGES are described, the information content of IGES is descussed, and the advanced features defined for IGES are presented.

Chapter two contains a description of the physical structure of an IGES file. It describes the rules for each of the sections in IGES, and the use of constants, strings, and free format.

Chapter three has five sections each of which presents a description of the data for a class of IGES entities. Included in chapter three are graphic examples selected to illustrate the entities being defined.

There are four appendices included in this reference manual. Appendix A describes the macro capability and was written by Neil Webber of NBS. Neil has also produced an interpreter written in FORTRAN which expands IGES macros into IGES entities. The interpreter will be published shortly as a separate report.

Appendix B contains a discussion of the mathematics used in the IGES spline entities, with several references to the underlying literature. This appendix was written by John Lewis of General Electric. John Lewis and David Moore (of Boeing) were activeiy involved in the selection of the spline entities to be included in IGES.

Appendicies C and D show examples using IGES and were prepared by Philip Kennicott, and Walt Braithwaite.

1.2 IGES Goals

In formulating the IGES presented in this report, the technical committee tried to achieve several goals. The overriding goal was to produce an initial specification early in 1980 to meet the deadline. It was felt that the time constraint was not a limiting factor in producing an acceptable specification, and a set of additional goals was formulated as follows:

1. To produce a format that would permit the communication of basic Geometry, Drafting, and structural entities.

2. To produce an open end format, to facilitate the communication of new material defined after the IGES was published.

3. To minimize, where possible, the burdens imposed on pre- and post-processors by the IGES.

4. To gather as much input from the interested community as possible.

The document contained in this report is the result of the technical committee's work over the past three months. In order to achieve these goals, extensive reviews of CAD/CAM systems were conducted. Several vendors and corporate system users were visited, and two public meetings were conducted.

An early decision was made to use the Boeing CAD/CAM Integrated Information Network (CIIN) standard format as the basis on which to define IGES. This was done because that system had been in use for a similar purpose for several years and was known to work. As a result the IGES structure is similar to that of CIIN. However, IGES is not purely a CIIN derivative. Many of the features included in IGES were found to be lacking in CIIN. The General Electric Neutral Data Base was used as a source of advanced concepts, as were numerous other formats provided to the technical committee.

The list of entities in IGES is divided into four categories. The entities in the categories of Geometry, Drafting, and Other were included to meet goal

number one. The Definition entities were included to meet goal number two. The defining of each of the entities in this document has undergone several iterations before the selection of the final definitions presented here. The revisions were the result of various forms of interaction both within the committee and with the interested public.

Goal number two was responsible for more than just the Definition entities mentioned above. The concept of extensibility was considered to be very important, and resulted in a number of features for future use. For example almost all entities in IGES can have pointers to properties and associativity entities. This was done because it is clear that this capability will be needed in the near future. It is recognized that in the absence of IGES defined properties and associativities, the current specification provides for their communication, but not their meaning. Thus IGES has provided the alphabet, but not the dictionary. Nevertheless, given that the alphabet exists, it is now possible for an IGES working committee, or some other group to write the dictionary. Similar arguments with respect to an alphabet in the absence of a dictionary can be made with respect to the macro definition entity. However, in the case of a macro, IGES does provide a default meaning by the expansion of the macro definition. It is worth noting that a macro expansion program for IGES macros has been written in FORTRAN, and will be distributed shortly.

Goal number three was a constant cause of discussion in IGES meetings. Often a feature was adopted or discarded on the basis of the thought experiment of mentally constructing an algorithm which could efficiently process the feature. The structure of the IGES file is an example.

IGES entities were divided into two sections. The first section, called a directory entry section, consists of fixed-field records, while the second section, called the parameter section, has variable length records and uses a free format. These decisions were made for simplification of the pre- and post-processors. Furthermore, as a general rule, the information that applies to all entities is placed in the directory entry section, while entity-specific information is in the parameter section. In addition, several fields of

information in the directory entry section were designed to simplify pre- or post-processing. Examples are the count of the number of parameter cards, the sub-entity flag, and the version number field. The form number field was created to allow several entities to share the same parameter field with a slight change in interpretation. These and many other features of IGES were dictated by goal number three.

The fourth goal was to gather input from sources outside the committee. This was done formally at two meetings, and informally in visits, phone calls, and letters. Although the specifics of this influence are not pointed out here, this process had a profound effect on the IGES as it appears here.

1.3 Information in IGES

The information in an IGES file is contained in the entities. An entity is a basic unit of information, and IGES has defined four broad categories of entities as follows:

1. Geometry
2. Drafting
3. Definition
4. Other

The list of entities in each category, and the specific data associated with each entity are found in Section 3. In this section, the overall structure of the entities and their interrelationships are described.

1.3.1 Entity Structure

Each entity in IGES has two parts. The first part consists of a directory entry, and the second contains parameter data. The directory entry section is fixed in length, while the parameter entry section has a variable length. This permits a uniform structure within the IGES file with a minimum of wasted space. The specific information in the directory section for an entity is the same for all entities and is described in section 2.3. The parameter section data is different for each entity and is described in Section 3.

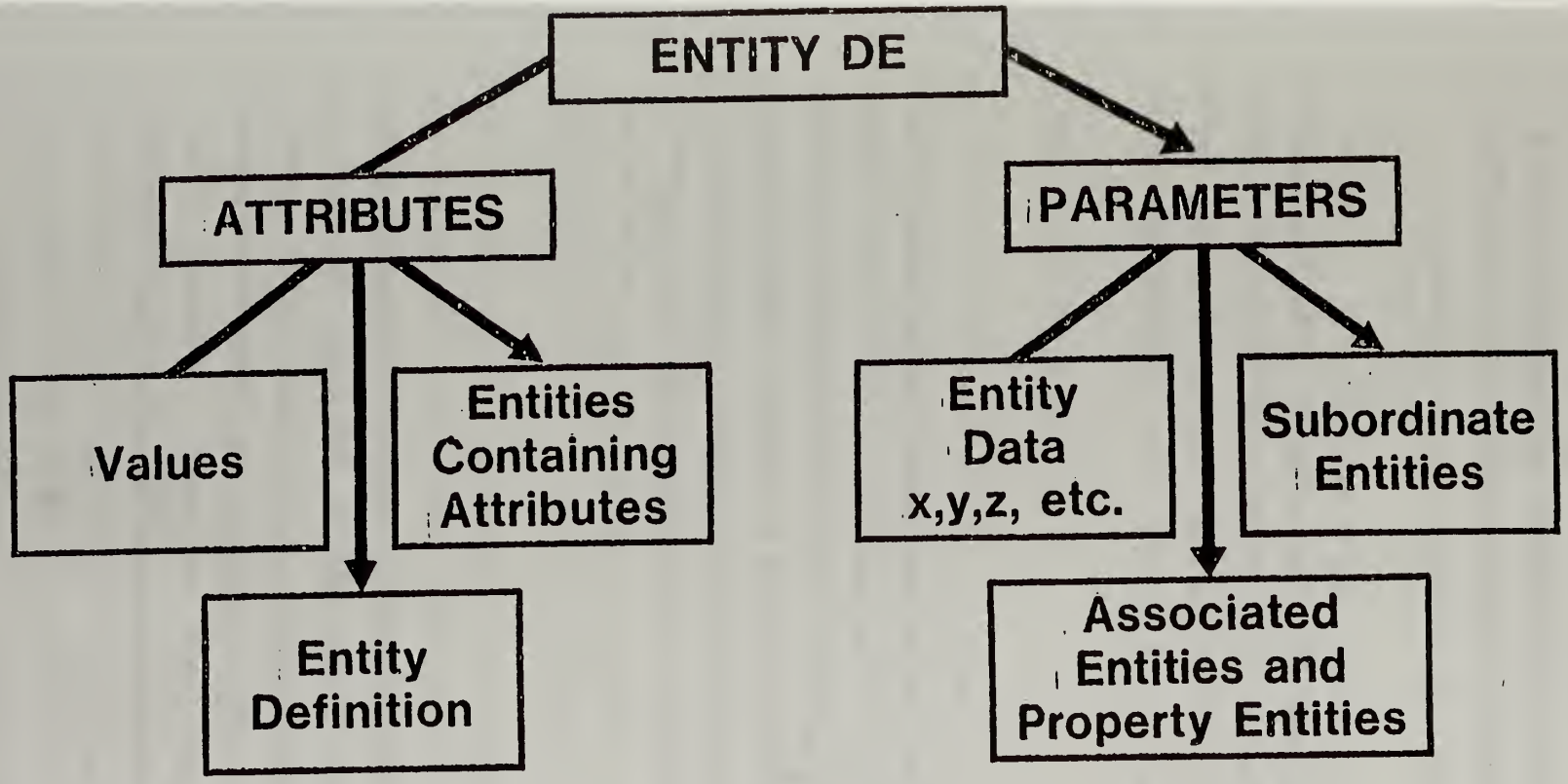

ENTITY DE
ATTRIBUTES
PARAMETERS
Values
Entities Containing Attributes
Entity Definition
Entity Data x,y,z, etc.
Subordinate Entities
Associated Entities and Property Entities

FIGURE 2

1.3.2 Pointers

IGES entities frequently refer to other entities in their definition. This is accomplished by the use of pointers to the Directory Entry (DE) of the referenced entity. See for example the definition of a linear dimension.

There are several categories of pointers used in the IGES file. All of the pointers are treated in the same way (i.e., point to the DE of the referenced entity), with the exception of the pointer in a DE to the parameter data. The pointer to parameter data is found in field two of the directory entry and refers to the parameter section of the file. The use of pointers in an entity DE is illustrated in Figure 2.

SUBORDINATE ENTITIES

An IGES entity can be considered a subordinate entity, if it was created to be a subpart of some other entity. A witness line pointed to by a linear dimension is an example of a subordinate entity. The concept which operates is as follows: if the entity was created purely as a subpart of another entity it is a subordinate entity and does not exist independently. Thus for example the members of a group pointed to by an associated entity are not subordinate entities. There is a flag in the DE block of each entity which indicates whether or not it is a subordinate entity.

ASSOCIATED ENTITIES

Most of the entities in IGES have a location in the parameter section which points to associated entities. These pointers can point to either an associativity entity of which the pointing entity is a member (i.e., it is a back pointer to an associativity), or to a general note entity. If the pointer is to a general note, then an associated text relationship is indicated. Future releases of IGES may expand this ability to include other implicit associations.

ASSOCIATED PROPERTIES

Most of the IGES entities can point to an arbitrary number of property entities from within their parameter sections. Each property entity has a form number to indicate its meaning. When an entity points to a property, it indicates that the information in the property is related to the entity.

ATTRIBUTE POINTERS

Within the DE block of each entity there are several fields that may be pointers. Each of these fields points to a specific type of attribute entity. The details of these references are presented in the description of the DE block in Section 2.3.

Advanced Features of IGES

In order that IGES may continue to develop in an orderly fashion several advanced features have been included within the specification. These are features whose support is only in a rudimentary form in existing commercially available interactive graphics sytems, but which, based on the philosophy of IGES, appear necessary to provide an orderly path for advancement of future IGES releases. The advanced features include two types of non-geometric data entities, the view entity, and the drawing entity.

Non-Geometric Data

There are two types of non-geometric data accommodated by IGES: that accommodated by the property entity and that accommodated by the associativity entity. The property entity is meant to collect non-geometric data of a general type which must be related to a specific entity. Any entity in IGES may point to one or more property entities which contain auxiliary information as necessary to adequately describe the entity.

Up to 9999 property types (form numbers) may be included in an IGES file. The lower half of these property numbers are reserved to the IGES specification and would be used for such purposes as, for example, specifying the width of a composite curve when it is to represent a wire run of a fixed width

on a printed wiring board. The remaining property form numbers are reserved for the user. The IGES specification cannot detail how a user's specification is to be handled on an arbitrary interactive graphics system beyond communicating the specified relationships between properties and entities.

An examination of the property entity will indicate that the property entity itself may point to other property entities, thus facilitating the construction of a non-geometric data network. Networks are useful for maintaining information such as signal strings, dimension dependencies, etc.

The concept of a network can be enhanced by the use of the associated entity pointers in the property entity. Using these pointers, properties (and their networks) can become members of associativities.

While the property entity is designed to accommodate non-geometric data associated with a specific geometric entity, the associativity entity is designed for use when several geometric entities must be logically related to each other. An example of such a relationship might be the association of four lines to indicate a rectangle. The implication is that the four lines are to be regarded as a single construct by the design system.

The reader will note that several of the functional capabilities of IGES have been implemented as associativities. For example, the group function is an associativity. This associativity asserts that the member entities are related to each other in some manner which need not be made clear by the system. The views visible associativity entity allows specification of display properties for a specific entity in a specific view. It is the fact that this information is both view-specific and entity-specific that makes the use of the associativity attractive.

As with properties, associativities can point to other associativities or to properties. If it is important that back pointers exist from an associativity to member entities, it is possible to indicate this requirement. The associativity entity may contain additional non-geometric information aside from the

pointers to member entities. This is useful when non-geometric information is more appropriately included in the associativity than in any member entity and when the amount of such information is small enough not to warrant the use of a property entity.

As with the property entity, the associativity entity has 9999 form numbers available. The lower half of these numbers are reserved to the IGES specification, while the upper half may be used as required by the user. In order to adequately indicate the desired handling of unknown (user-specified) associativities by a post-processor, the associativity definition is included. Support by a post-processor of the associativity definition implies that links will be maintained in the target system database as specified in the definition.

The associativity definition allows the user to specify multiple classes within an associativity. This is useful, for example, when there are several types of relationships to be maintained which are logically related to each other. It avoids the use of multiple entities. An example is a signal string in an electrical design. Here, one class of associated entity represents the logical signal string with its contact nodes, a second class represents the physical layout of the string on a schematic logic diagram, and a third class represents actual wiring runs on a printed circuit wiring board.

Each class is defined independently. The user is allowed to specify whether or not back pointers are required. If back pointers are required, all entities that are members of the associativity must have pointers to the associativity in their parameter sections. This facilitates the transmission of the correct data structure to a post-processing system. The list of entities in a class may be specified as either ordered or unordered. In some cases (e.g., the group) the order in which member entities appear in a list is irrelevant, while in other cases (e.g., the four lines representing a box) the order is important and must be maintained by a supporting system. Within each class the number and type (i.e., integer, character string, or pointer) of each item in each entry may be specified. The integer and character string allow inclusion of a limited amount of non-geometric information in the entry.

1.4.2 Views

There are two often confused, and sometimes conflicting, goals of interactive graphics systems: to represent the geometry of an object or model, and to present this geometry to a human in an understandable and recognizable form. The same conflict exists in a communication medium designed to transfer information between systems. The view entity allows the separation of these two functions. On an interactive graphic system it permits the designer to create the geometry independently of any concern as to how it should best be presented to other humans. As he creates the geometry, the designer makes full use of the viewing capabilities available on his system. After creation of the geometry, the view entity provides the user with a method for collecting together the necessary information to present one "picture" of the object. Incorporation of the view entity in IGES provides the means for communicating this collection structure to another graphics system.

The view specifies the view point (i.e., the position of an imaginary eye). This is sometimes confused with a rotation of the object, but it should be kept in mind that the object is maintained stationary in model space, while the eye position is being moved.

Often it is desirable to present only a portion of an object as, for example, in a detail on a mechanical drawing. Such a capability requires the addition of a clipping box and a scale factor. This allows extraction of a portion of the model geometry with ends of lines being clipped in a satisfactory manner. After extraction of the desired portion of the model, the scale factor allows the display of the portion of the geometry at any desired magnification for better understanding of detail. This order of application of clipping and scaling is taken so as to allow the use of features in the model to define the clipping box. The alternative of scaling first and then clipping, as is more common in other computer graphics applications, was discarded.

Often a set of entities must be displayed in a given view with particular display characteristics. An example is the treatment of lines which should be

hidden in a particular view. Should they be shown as is, blanked out, or dashed? The views visible associativity, together with the view entity, allows such a specification to be made and communicated to a new host system.

Drawing

The drawing entity has been defined in order to provide a two dimensional surface on which a set of views can be projected and arranged in a manner which is meaningful and informative. Several views may be placed on a drawing in any arrangement the user desires. In addition, drafting entities may be added to provide additional information such as dimensions, notes, etc., in a manner similar to the use of such information in a conventional mechanical drawing.

The philosophy behind the incorporation of the view and drawing entities in IGES is the principle that a specific piece of information should be held in a single place in a design data base and that, if it is changed, the effect of this change should become immediately apparent whenever the information is used. While this is a design data base principle, it is equally important in a transmission format for communicating the information in the data base.

The introduction of the view and drawing entities allows changes made in the geometry of a part to be reflected in each view of the part, thus minimizing the danger that differing versions of a part will coexist.

It would also be desirable for the drafting entities to be automatically updated when changes are made in the model. Such a capability would result, for example, in dimension values changing to reflect changes in the physical shape of the model. Unfortunately, this is only possible in special circumstances; more research is required to make it happen in the general case. When this research is done, the result will be a set of associativities relating drafting and geometrical entities so as to reflect the desired changes in the drafting entities in relation to changes in the geometrical entities. The capability exists in the IGES specification to communicate these associativities when they have been defined.

1.4.4 Summary

Several advanced features of IGES have been described in an attempt to outline both the intended function of the advanced entities as well as the philosophy governing their design. It is hoped that developers of interactive graphics systems can use these features in furthering the state-of-the-art of interactive graphics while staying within the philosophy adopted for the exchange of graphics data within IGES.

IGES FILE STRUCTURE

An IGES file is written on 80 column card images, using the ASCII (7-bit) character set. The file contains five subsections which must appear in order as follows:

1. Start Section
2. Global Section
3. Directory Entry Section
4. Parameter Data Section
5. Terminate Section.

Each card image in the file has a unique letter in column 73 which identifies the section to which it belongs, followed by a sequence number in columns 74 through 80 to indicate its position within the section. The sequence number for each section begins with 0000001 and ends at the appropriate number for the section.

Leading zeros in the sequence field may be optionally left as blanks. The letter codes for column 73 are as follows:

SECTION		LETTER CODE
1.	Start	S
2.	Global	G
3.	Directory Entry	D
4.	Parameter Data	P
5.	Terminate	T

Start Section

The start section of an IGES file is designed to provide a man-readable prolog to an IGES file. There must be at least one start card, and all cards in the section must use the sequence number field in column 73 through 80. Column 73 must contain the letter S and the sequence numbers must begin with 1.

START SECTION

THIS SECTION IS A MAN READABLE	S0000001
PROLOG TO AN IGES FILE. IT CAN CONTAIN	S0000002
AN ARBITRARY NUMBER OF CARDS	S0000003
⋮	⋮
USING ASCII CHARACTERS IN COLUMNS 1-72	S0000020

The information in columns 1 through 72 is not formatted in any special way except that the ASCII character set must be used. An example of a start section is shown in Figure 3.

Global Section

The global section of an IGES file contains the information describing the pre-processor and information needed by the post-processor to handle an IGES file. Sequence numbers in the global section have a "G" in column 73 and begin with number 1. The first two global parameters are used to redefine the delimiter and end of line characters if necessary. The default characters are "Comma and Semicolon" respectively.

The parameters for the global section are input in free format as described in section 2.7. If the global section specifies new delimiter characters, they take over immediately and are used in the Global section as well as the rest of the file. This is possible, because the comma and semicolon delimiter functions are the first two global parameters. It is expected that if the comma and semicolon delimiters are not to be changed, the global section will begin with ",," to indicate the default values are desired.

The parameters in the Global Section are described in Figure 4.

FIGURE 4

PARAMETER	FIELD TYPE	DESCRIPTION
1	Text	Delimiter character (default=,)
2	Text	End of line character (default=;)
3	Text	Drawing identification
4	Text	File name
5	Text	System ID . Vendor . Software version
6	Text	IGES translator version
7	Integer	Number of bits for integer representation
8	Integer	Number of bits in a single precision floating point exponent
9	Integer	Number of bits in a single precision floating point mantissa
10	Integer	Number of bits in a double precision exponent
11	Integer	Number of bits in a double precision mantissa
(TO SYSTEM)		
12	Text	Drawing identification
(FILE INFORMATION)		
13	Floating point	Drawing scale
14	Integer	Unit 1, inch
15	Text	2, mm 3, other
16	Integer	Maximum line weight (0-512)
17	Floating point	Size of maximum line width in 'units'
18	Text	Date & time of file generation 'XXYYZZ,AABBCC'
19	Integer	Smallest gridsize of definition space
20	Integer	Size of definition space
21	Text	Name of author
22	Text	Organization

Directory Entry Section

The directory entry section of an IGES file has one entry for each entity in the IGES file. The directory part of each entity is fixed in size, and contains twenty fields of eight columns spread across two cards for each entity. Note, all DE pointers are odd valued. The purpose of the directory entry is to provide an index for the file and to contain attribute information. The directory entry has either attribute values directly specified in a specific field, or a pointer to a location in the file where the information is located. Some of the fields in the directory entry can contain an attribute value directly, or a pointer to a set of such values. In these fields a negative number indicates a pointer, while a positive number is taken to be the attribute value. In the following paragraphs each of the directory fields is described. Figure 4 shows the layout of a directory entry and has a number sign or arrow to indicate the type of data which appears in the field. When a field in Figure 5 has both a number sign and an arrow either type data is possible, and two descriptions of the field are included.

DIRECTORY ENTRY FIELD DESCRIPTION

NO.	FIELD NAME	MEANING AND NOTES
1	Entity type No. #	Identifies the IGES entity No.
2.	Parameter Data =>	A pointer to the first card of the Parameter Data for the entity.
3	Version #	A version No. which indicates how to interpret the Parameter Data for this entity. This value will be 1 for all entities in the initial release of IGES.
3	Version =>	A pointer to the DE of the definition entity that specifies this entity's meaning.
4	Line Font Pattern #	Selection of a system line font. 1 Solid 2 Dashed

		3 Phantom 4 Centerline
4	Line Font	A pointer to the DE of a line font pattern entity.
	Pattern =>	
5	Level #	Entity is defined on this level
5	Level =>	Pointer to the DE of an associativity entity which contains a list of levels on which the entity is defined.
6	View =>	Pointer to the DE of a view entity or to views visible associativity entity.
7	Defining Matrix =>	Pointer to the DE of a matrix used in defining this entity; zero implies the identity matrix will be used.
8	Label Display Associativity=>	Pointer to the DE of a label display associativity (Form 5).
9	Status #'s	Provides four two digit status values. The first three are used; the fourth is reserved. 1-2 Blank Status 00 Visible 01 Blanked 3-4 Subentity switch 00 Independent 01 Dependent 5-6 Geometry Flag 00 Geometry 01 Drafting 02 Definition 03 Other 7-8 Unused
10	Sequence #	Physical count of this record from the beginning of the Directory Entry section, preceded by the letter D (odd number).
11	Entity type #	(Same as Field 1)
12	Line Weight #	System display thickness; normal is zero.

DIRECTORY ENTRY (DE) SECTION

CARD 1	ENTITY TYPE NO.	PARA-METER DATA	VERSION	LINE FONT PATTERN	LEVEL	VIEW	DEFINING MATRIX	LABEL DISPLAY	STATUS	SEQ #
	#	➧	#, ➧	#, ➧	#, ➧	➧	➧	➧	#'s	D-------
	1	2	3	4	5	6	7	8	9	10
CARD 2	ENTITY TYPE NO.	LINE WEIGHT	PEN NUMBER	PARAMETER CARD COUNT	FORM NUMBER	RESERVED	RESERVED	ENTITY LABEL CHARAC-TERS	ENTITY SUBSCRIPT	SEQ #
	#	#	#	#	#				#	D-------
	11	12	13	14	15	16	17	18	19	20

- Number

➧ - Pointer

#, ➧ - Number or pointer (pointer has NEG sign)

13	Pen Number #	Pen or color number.
14	Parameter card Count #	Number of cards in the Parameter Data for this entity
15	Form Number #	Attaches a form number to the entity
16-17	Reserved for future use	
18	Entity Label	Up to eight alphanumeric characters (right justified).
19	Entity subscript#	1 to 8 Digit unsigned number associated with the label
20	Sequence #	Same meaning as field 10 (even number)

2.4 Parameter Section

The parameter section of an IGES file contains the parameter data associated with each entity. The specific format of the parameter data for each entity is described in Section 3.0. The following information is true for all parameter data.

Parameter data is placed in free format (see section 2.7) with the first field always containing the entity type number. Therefore, the entity type number and a comma precede parameter one of each entity. The free field part of the parameter card ends in column 64. Columns 65 through 72 on all parameter cards contain a pointer to the sequence number of the first card in the directory entry for the entity for which parameter data is being generated. Columns 73 through 80 contain a parameter sequence number with a "P" in column 73 and a sequence number in columns 74 through 80. The sequence number begins with a 1 on the first parameter card for the first entity in the parameter section. With the exception of text strings, all parameter values are restricted from crossing card boundaries. Thus, numeric values must start and end on the same card. When a text field does cross card boundaries, column 64 on the current card is considered to be next to column 1 on the next card. Parameters to be defaulted are indicated by

PARAMETER DATA SECTION

	65	73
CARD 1 ENTITY TYPE NO., (Parameters separated by commas)	DE PTR	SEQUENCE NUMBER P #
CARD 2 (Parameters separated by commas)	DE PTR	SEQUENCE NUMBER P #
CARD 3 ... • • • •		
CARD N (Parameters separated by commas) ;	DE PTR	SEQUENCE NUMBER P #

DE PTR POINTS TO THE DIRECTORY ENTRY FOR THIS ENTITY
SEQUENCE NUMBER BEGINS WITH THE LETTER P AND IS SEQUENTIALLY NUMBERED

two commas with zero or more intervening blank characters. A semicolon indicates that the parameter list is complete and any remaining parameters should receive default values. Semicolon should always be the last character of a parameter set, even if all parameters were explicitly specified.

Any desired comment may be added after the semicolon. Note that additional cards may be used for this purpose by keeping the DE pointer in columns 65-72 constant.

The default value for a numeric argument is zero and for a text argument is a null string. It is the responsibility of the preprocessor which creates an IGES file to make sure that the default value is a reasonable one for the particular parameter. Figure 6 shows a parameter section.

2.5 Terminate Section

There is only one card in the terminate section of an IGES file. It is divided into ten fields of eight columns each. The terminate section must be the last card of the IGES file. It has a "T" in column 73 and 0000001 in columns 74 through 80.

The fields on the terminate card contain the last sequence number used in each of the previous sections defined below, see Figure 7.

FIELD	COLUMNS	SECTION
1	1-8	Start Section
2	9-16	Global Section
3	17-24	Directory Entry Section
4	25-32	Parameter Section
5-9	33-72	(not used)
10	73-80	Terminate Section

TERMINATE SECTION

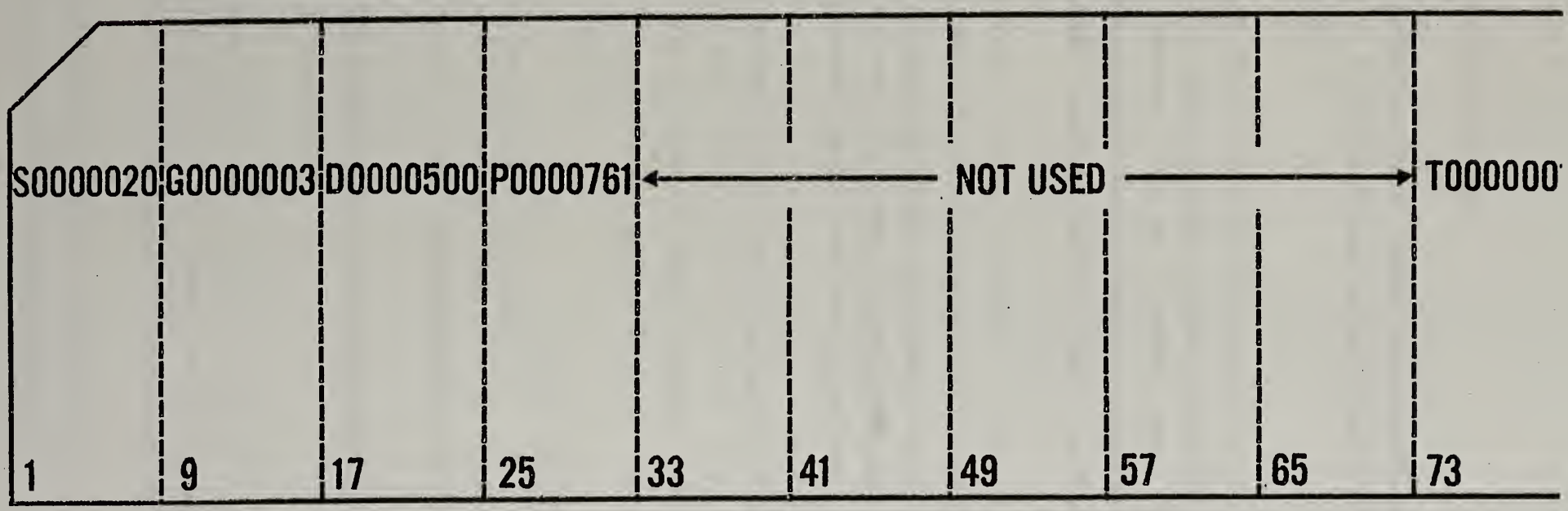


FIGURE 7

2.6 Constants in IGES

IGES defines three types of constants, integer, real, and string. The acceptable forms of each are defined in the sections that follow.

2.6.1 Integer Constants

An integer constant is composed of one or more numerical characters. Although formally called an integer constant, it is more commonly called a fixed-point or integer number because of the fact that the decimal point is always assumed to be located to the right of the last numerical character of the number.

An integer constant may be either positive, zero, or negative; while a positive integer number can have the special character plus (+) as its leading character, if an integer number is unsigned and nonzero IGES assumes it to be positive. An integer number must comply with the following four rules:

1. It must be a whole number. That is, it cannot contain a decimal point.

2. If negative, the special character minus (-) must be the leading character.

3. It cannot contain embedded commas.

4. Its maximum magnitude can be either plus or minus 2**(N-1) -1 (where N is parameter seven from the global section).

The following are examples of valid integer constants (assuming N is 32).

```
1
150
2147483647
0
-10
-2147483647
```

.2 Floating Point Constants

IGES permits both single and double precision floating point constants. The precision of these constants is specified in the Global section, in parameters 8 through 11.

A single precision floating point number may be expressed with or without an exponent. Double precision constants must be in exponential form.

A floating-point constant without an exponent is composed of one or more numerical characters and the special character period (.) that may be followed by one or more of these numerical characters to form what is called the fractional part of the constant. Sometimes called a real constant, it is more commonly called a floating-point constant to reflect the fact that the decimal point can be moved or floated to either the beginning, middle, or end of the numerical characters forming the number. Floating-point constants may be either positive, zero, or negative. A positive floating-point constant can have the special character plus (+) as its leading character, if a floating-point constant is unsigned and nonzero IGES assumes it to be positive. A floating-point constant must comply with the following four rules:

1. If negative, the special character minus (-) must be the leading character.

2. It must contain a decimal point.

3. It cannot contain embedded commas.

4. The size of the number must be compatible with the parameters in the global section.

A floating-point constant may be expressed in exponential form. Single precision floating point numbers use the letter "E" in the exponent, while double precision floating point constants use the letter "D" in the exponent.

A floating-point constant in exponential form begins with a constant (real or integer) followed by an exponent letter ("E" or "D") followed by an integer

constant. The first constant is called the mantissa and the second constant the exponent. The value of the resultant floating-point constant is the value of the mantissa multiplied by ten raised to the power specified in the exponent. The precision of allowable numbers for the mantissa and exponent are given in the global section. Examples of floating point constants are below:

Single precision non exponent form:

```
264.091
0.
-.58
+4.21
```

Single precision exponent form:

```
1.36 E 01
12.943E1
-13.09E-2
123.409E-4
0.1E-3
1.0E+4
```

Double precision exponent form:

```
145.98763D+04
2145.980001D-5
0.123456789D 9
```

Note: Double precision floating point constants must use the exponential form.

String Constants

A string constant in IGES uses the Hollerith form as found in the ANSI specification of FORTRAN. A string constant is preceded by an unsigned integer, and the letter "H". String constants have the following rules:

1. The string is preceded by a count of characters and the letter "H".

2. Any character from the ASCII set may appear in the string. (Blanks, commas, and semicolons have no special meaning within a string constant.)

3. String constants may cross card boundaries in an IGES file (other constants may not). When a string constant does cross a card boundary, the last usable column on the current card is concatenated with column one on the succeeding card. (Note: the last usable column on parameter cards is column 64; on other cards it is column 72.)

4. There is no limit on the size of a string constant.

Examples of valid string constants are:

```
3H123
10HABC.,:ABCD
12H HELLO THERE
8H0.457E03
```

Free Format Rules

The data in several sections of an IGES file may be entered in free format. The free format feature allows the specification of parameters in a prescribed order, but does not specify a location on the card image. When free format is permitted the following rules apply:

1. Blanks are ignored.

2. Commas (,) are used to separate parameters.

3. Semicolon (;) is used to terminate the list of parameters.

4. When two commas appear adjacent to each other (or separated only by blanks) the parameter is not specified in the file and should be given a default value.

5. If a semicolon appears before the list of parameters is complete, all remaining parameters should be given default values.

6. Blanks are not ignored in string constants. In addition the comma and semicolon are treated as characters in a string constant and do not have the meaning specified in (2 through 5).

3.0 DEFINITION OF ENTITY FORMATS

This section of the document provides the standard format for each entity component.

Each entity has two basic parts to its representation; the Directory Entry (DE) and the Parameter Data Entry (PDE). This section contains pertinent information needed for the directory entry, and a description of the PDE portion of the representation.

The directory data section of each entity will have codes showing what information must be present in the directory entry. These attributes were explained in the previous section, the codes and meanings are listed below.

MTX: Matrix reference must be present

FORM: The form number of the entity

3.1 Geometry Entities

The following geometric entities are defined in this section:

Entity Name	Entity Number
Circle	100
Composite Entity	102
Conic	104
Copious Data Block	106
Face	108
Line	110
Parametric Spline	112
Parametric Bicubic Spline Surface	114
Point	116
Ruled Surface	118
Surface of Revolution	120
Tabulated Cylinder	122
Transformation Matrix	124

CIRCLE

The arc of a circle is defined in an XT-YT plane different from model space. A transformation matrix is specified for transforming the XT-YT space into model space. The circle is defined in terms of end points and the arc center coordinates; the points are defined so that the desired arc is traced by moving from point one to point two in a counterclockwise direction.

In example 3 of Fig. 3.1-1 the solid arc is defined using point A as the first end point and point B as the second end point. If the dashed arc were desired, the first end point would be B and the second end point A.

Examples of circle entities are shown in Fig. 3.1-1.

Directory Data

ENTITY NUMBER : 100
ATTRIBUTES REQUIRED : MTX

Parameter Data

Parameter	Value	Format	Comment
1	z	Floating Point	Center displacement from XT-YT plane
2	x	Floating Point	Circle center abscissa
3	y	Floating Point	Circle center ordinate
4	x	Floating Point	End point one abscissa
5	y	Floating Point	End point one ordinate
6	x	Floating Point	End point two abscissa
7	y	Floating Point	End point two ordinate

Parameter	Value	Format	Comment
8	N	Integer	Number of associate entities
9	DE	Integer	Pointers to DEs o associated entities
.	.	.	
.	.	.	
.	.	.	
8+N	DE	Integer	
9+N	M	Integer	Number of associate properties
10+N	DE	Integer	Pointers to DEs o associated properties
.	.	.	
.	.	.	
.	.	.	
9+N+M	DE	Integer	

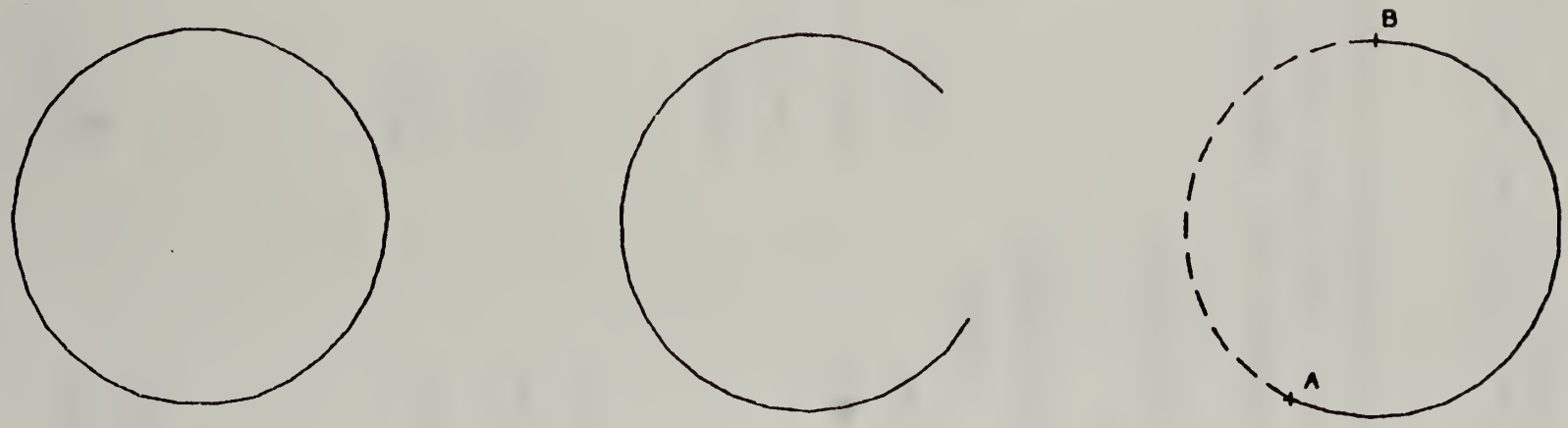


FIG. 3.1-1 EXAMPLES OF CIRCLES

COMPOSITE ENTITY

The composite entity is a concatenation of point, lines, arcs, conics, and parametric splines.

Each entity may have its own matrix, level, and line type. By definition, the end point of each entity is the start of the next. Each entity may have text or properties associated with it. An example of a composite entity is shown in Fig. 3.1-2.

Directory Data

ENTITY NUMBER : 102

Parameter Data

Parameter	Value	Format	Comment
1	N	Integer	Number of entities
2	DE	Integer	Pointer to Directory
.	.	.	Entries for the associated
.	.	.	entities
.	.	.	
N+1	DE	Integer	
N+2	NA	Integer	Number of associated entities
N+3	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
N+2+NA	DE	Integer	
N+3+NA	M	Integer	Number of associatec properties

Parameter	Value	Format	Comment
N+4+NA	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
N+3+NA+M	DE	Integer	

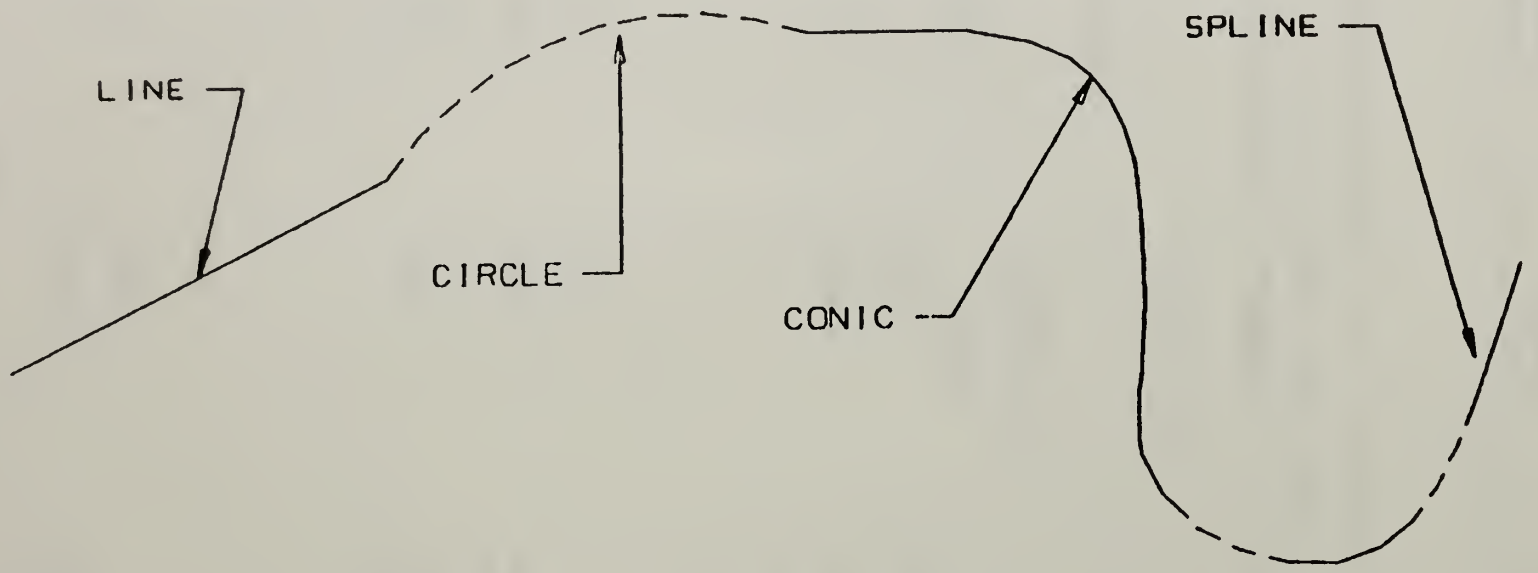


FIG. 3.1-2 EXAMPLE OF THE COMPOSITE ENTITY

CONIC

The general format of a conic equation is $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$. A conic can be defined in any XT-YT plane. The transformation matrix transforms XT-YT space into model space. The conic is defined by its six coefficients, a start, and an end point. Start and end points (P1, P2) are selected for ellipses such that a line from the conic center to end point 1 rotated in a counterclockwise direction toward point P2 will intersect the conic along the desired segment. The conic center is the intersection of the conic's major and minor axis (the dashed lines in Fig. 3.1-3). For hyperbola, the start and end points determine the branch of the hyperbola. An example of each conic type is shown in Fig. 3.1-3.

Within the conic Directory Entry the curve type is indicated by the Form number, which has the following meanings:

FORM	Meaning
0	Conic form must be determined from conic equation
1	Conic is Ellipse (See Example 1, Fig. 3.1-3)
2	Conic is one branch of a Hyperbola (Example 2, Fig. 3.1-3)
3	Conic is Parabola (Example 3, Fig. 3.1-3)

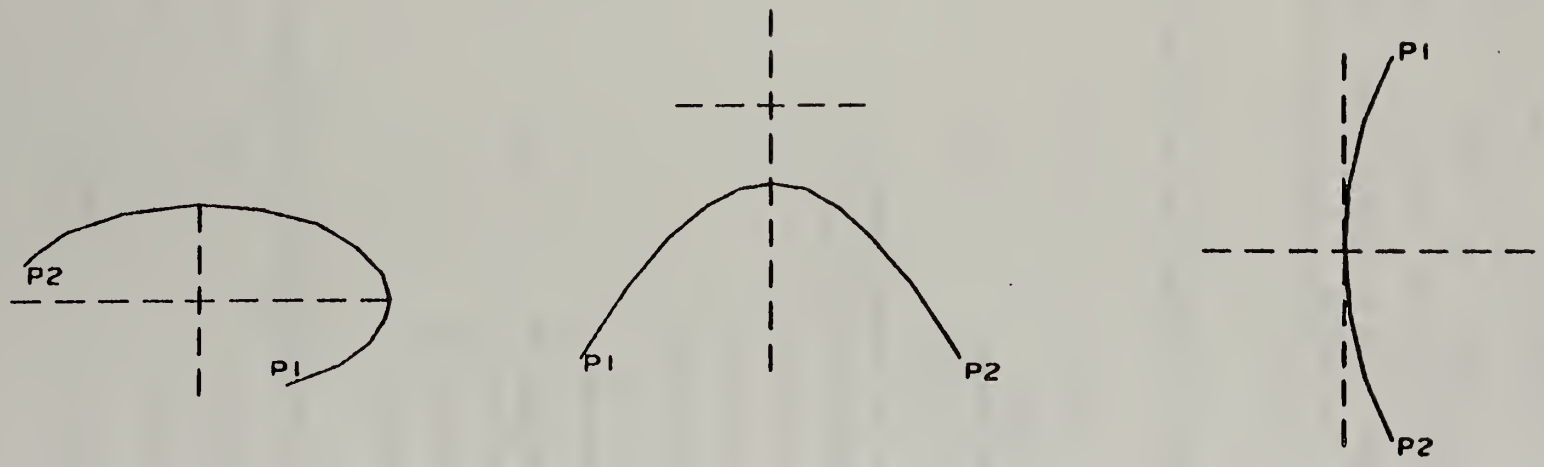
Directory Data

ENTITY NUMBER : 104
ATTRIBUTES REQUIRED : MTX, FORM

Parameter Data

Parameter	Value	Format	Comment
1	A	Floating Point	Conic Coefficient
2	B	Floating Point	Conic Coefficient
3	C	Floating Point	Conic Coefficient

Parameter	Value	Format	Comment
4	D	Floating Point	Conic Coefficient
5	E	Floating Point	Conic Coefficient
6	F	Floating Point	Conic Coefficient
7	Z	Floating Point	Conic Displaceme from XT-YT plan
8	X1	Floating Point	Start Point Abscis
9	Y1	Floating Point	Start Point Ordin
10	X2	Floating Point	End Point Absicis
11	Y2	Floating Point	End Point Ordinat
12	N	Integer	Number of associ entities
13	DE	Integer	Pointers to DEs associated entitie
.	.	.	
.	.	.	
.	.	.	
12+N	DE	Integer	
13+N	M	Integer	Number of associ properties
14+N	DE	Integer	Pointers to DE associated proper
.	.	.	
.	.	.	
.	.	.	
13+N+M	DE	Integer	



EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.1-3 EXAMPLES OF CONICS

COPIOUS DATA BLOCK

This entity stores data points in associated pairs, tuples, or sextuples. The interpretation flag value determines whether the storage is in pairs, tuples, or sextuples. The 2-tuples represent x,y pairs with a common z depth. The 3-tuples are x,y,z coordinates. The 6-tuples are x,y,z coordinates with an i,j,k normal vector at the coordinate.

The form number in the Directory entry will indicate the use of the copious data for specific entities.

Form Number	Entity Use
0	Copious Data
10	Linear string
20	Centerline through points
21	Centerline through lines
31	Section form 31
32	Section form 32
33	Section form 33
34	Section form 34
35	Section form 35
36	Section form 36
37	Section form 37
38	Section form 38
40	Witness line

Refer to section 3.2 for examples of the Centerline, Section, and Witness line entities.

Directory Data

ENTITY NUMBER : 106
ATTRIBUTES REQUIRED : FORM

Parameter Data

Parameter	Value	Format	Comment
1	IF	Integer	Interpretation flag IF=1; x, y pairs, common z IF=2; x, y, z coordinates IF=3; x, y, z coordinates and i, j, k normal vector
2	N	Integer	Number of 2 - tuples 3 - tuples or 6 - tuples
3	Data	Floating Point	If IF=1, K=3+2N the third word is a Z displacement if:
.	Points		IF=2, K=2+3N
.			IF=3, K=2+6N
.			
K			
K+1	N	Integer	Number of associated entities
K+2	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.		
K+1+N	DE	Integer	
K+2+N	M	Integer	Number of associated properties
K+3+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
K+2+N+M	DE	Integer	

FACE

The face entity represents either an unbounded plane or a bounded face. The plane equation Ax + By + Cz - D=0 defines the surface in either representation. The boundary of a face is represented by a composite entity. A center location and size parameter is used to control a system dependent display symbol for the unbounded plane. If the entity is a bounded face, parameters 6 through 9 are not included in the definition. Examples of the face entity are shown in Fig. 3.1-4.

The form number located in the Directory Entry will indicate if the face represents a positive face or a negative face (hole).

Form	Meaning
+1	positive face
-1	negative face

Directory Data

ENTITY NUMBER : 108
ATTRIBUTES REQUIRED : FORM

PARAMETER DATA

Parameter	Value	Format	Comment
1	A	Floating Point	Coefficients of Plane
2	B	Floating Point	
3	C	Floating Point	
4	D	Floating Point	
5	PTR	Integer	Pointer to DE o composite entity or 0
6	X	Floating Point	X coordinate of the default display symbol
7	Y	Floating Point	Y coordinate of the default display symbol
8	Z	Floating Point	Z coordinate of the default display symbol

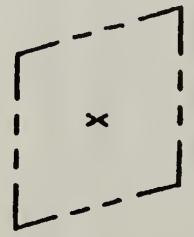
Parameter	Value	Format	Comment
9	SIZE	Floating Point	Height of the default display symbol
10	N	Integer	Number of associated entities
11	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
10+N	DE	Integer	
11+N	M	Integer	Number of associated properties
12+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
11+N+M	DE	Integer	

EXAMPLE 1

(UNBOUNDED)

EXAMPLE 2

(BOUNDED)



EXAMPLE 3

(UNBOUNDED)

FIG. 3.1-4 EXAMPLES OF THE FACE ENTITY

LINE

A line segment is defined by two end points (3-tuples) in model space. The line is considered to be defined from (X1, Y1, Z1) to (X2, Y2, Z2). Examples of line entities are shown in Fig. 3.1-5.

Directory Data

ENTITY NUMBER : 110

Parameter Data

Parameter	Value	Format	Comment
1	X1	Floating Point	Start Point
2	Y1	Floating Point	P1
3	Z1	Floating Point	
4	X2	Floating Point	End Point
5	Y2	Floating Point	P2
6	Z2	Floating Point	
7	N	Integer	Number of associated entities
8	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
7+N	DE	Integer	

Parameter	Value	Format	Comment
8+N	M	Integer	Number of associ properties
9+N	DE	Integer	Pointers to DEs associated proper
.	.	.	
.	.	.	
.	.	.	
8+N+M	DE	Integer	

EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.1-5 EXAMPLES OF LINES

PARAMETRIC SPLINE

The parametric spline is represented as a sequence of parametric cubic polynomial segments. It can also represent piecewise linear, modified Wilson-Fowler, Wilson-Fowler, and "B-Spline" Splines.

The N polynomial segments of the spline are delimited by the breakpoints t(1), . . , t(N+1). The coordinates of the points in the ith segment of the spline are given by the cubic polynomials:

$$X(u) = AX(i)+BX(i)*s+CX(i)*s^2+DX(i)*s^3$$
$$Y(u) = AY(i)+BY(i)*s+CY(i)*s^2+DY(i)*s^3$$
$$Z(u) = AZ(i)+BZ(i)*s+CZ(i)*s^2+DZ(i)*s^3$$

where

$$t(i) \leq u \leq t(i+1),\ i=1,N$$
$$s = u - t(i).$$

If the dimension of the spline is 2D, the coefficients of the Z polynomial will be zero. To enable rapid determination of the endpoint and derivatives without computing the polynomials, a dummy N+1st polynomial segment is included in the entity. The parameter t(n+2) is not provided for this segment since the endpoint of the dummy segment and the derivatives at that point are implied by the N+1 segment coefficients.

There is a parameter H which specifies the degree of continuity at the break points (H=2 for continuous second derivatives).

An example of a parametric spline is shown in Fig. 3.1-6.

Additional examples are shown in Fig. 3.1-7.

Directory Data

ENTITY NUMBER : 112

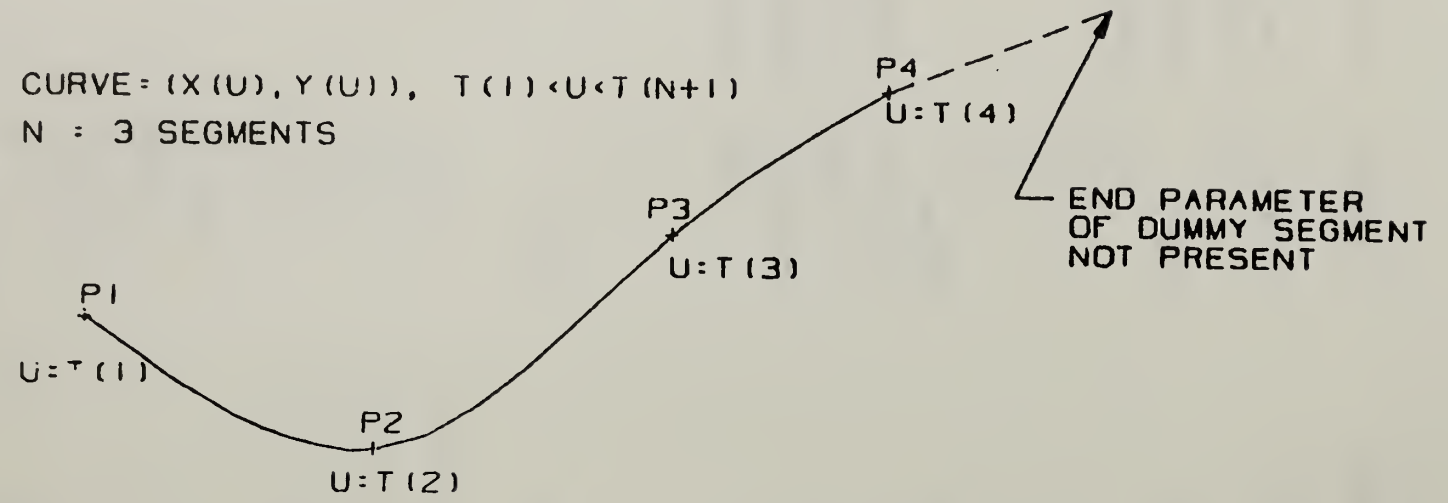
Parameter Data

Parameter	Value	Format	Comment
1	CTYPE	Integer	Spline Type (1=Linear 2=Quadratic 3=Cubic 4=Wilson-Fowler 5=Modified Wilson-Fowler)
2	H	Integer	Continuity
3	NDIM	Integer	Number of Dimensions (2=2D, 3=3D)
4	N	Integer	Number of Segments
5	t(1)	Floating Point	Break points of piecewise polynomial
.	.		
.	.		
.	.		
5+N	t(N+1)		
6+N	AX(1)	Floating Point	X Coordinate Polynomial
7+N	BX(1)		

Parameter	Value	Format	Comment
8+N	CX(1)		
9+N	DX(1)		
10+N	AY(1)		Y coordinate
11+N	BY(1)		polynomial
12+N	CY(1)		
13+N	DY(1)		
14+N	AZ(1)		Z coordinate
15+N	BZ(1)		polynomial
16+N	CZ(1)		
17+N	DZ(1)		
	.		Subsequent X, Y, Z
	.		Polynomials
	.		
6+13*N	AX(N+1)	X value at endpoint	Last polynomial
	BX(N+1)	X slope at endpoint	(A dummy
	CX(N+1)	X second derivative	segment included
	DX(N+1)	X third derivative	only to indicate
			the value of the
	AY(N+1)		spline and its
	BY(N+1)		derivatives at
	CY(N+1)		the endpoint)
	DY(N+1)		
	AZ(N+1)		Z polynomial
	BZ(N+1)		
	CZ(N+1)		
	DZ(N+1)		

TOTAL ENTRIES (TE) = 5+N+12*(N+1)

Parameter	Value	Format	Comment
TE+1	NA	Integer	Number of associated entities
TE+2	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
TE+1+NA	DE	Integer	
TE+2+NA	M	Integer	Number of associated properties
TE+3+NA	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
TE+2+NA+M	DE	Integer	



FOR SEGMENT NUMBER 2:

$$X(U) = AX(2) + BX(2) * (U - T(2)) + CX(2) * (U - T(2))^2 + DX(2) * (U - T(2))^3$$

$$Y(U) = AY(2) + BY(2) * (U - T(2)) + CY(2) * (U - T(2))^2 + DY(2) * (U - T(2))^3$$

P1 = (AX(1), AY(1))

P2 = (AX(2), AY(2))

P3 = (AX(3), AY(3))

P4 = (AX(4), AY(4))

DERIVATIVES AT P4 = (BX(4), BY(4))

FIG. 3.1-6 EXAMPLE OF PARAMETRIC CUBIC SPLINE (2D)

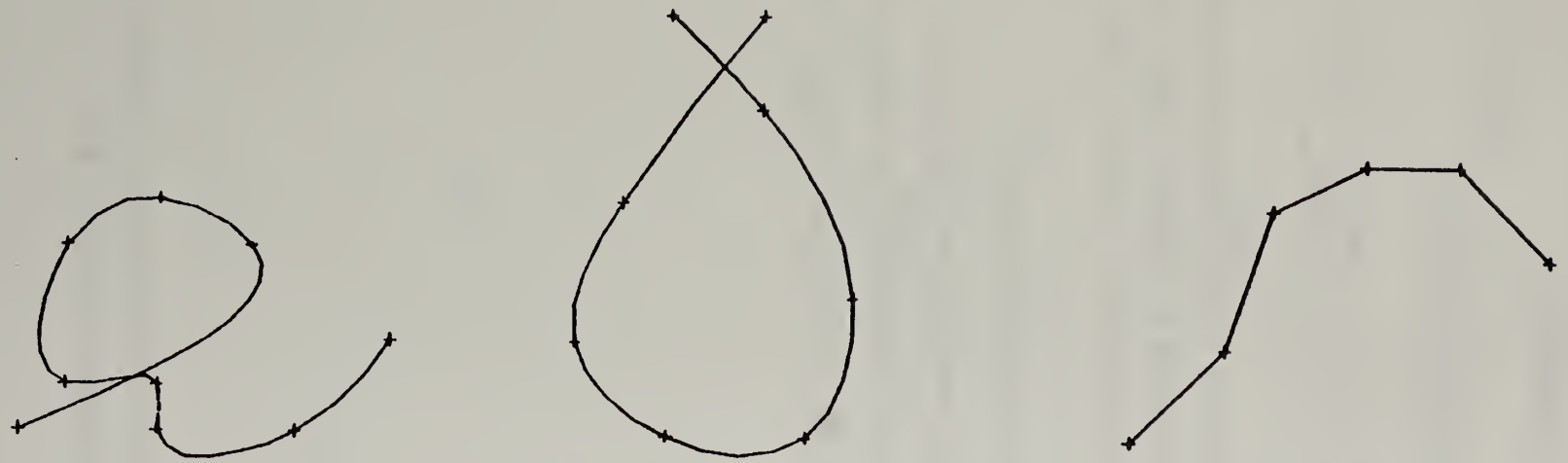


FIG. 3.1-7 EXAMPLES OF PARAMETRIC SPLINES

PARAMETRIC

BICUBIC SPLINE SURFACE

The parametric bicubic spline surface is a grid of parametric bicubic patches. It can represent bicubic patches and various cartesian product surfaces.

The MxN grid of patches is defined by the u breakpoints tu(1),...,tu(M+1) and the v breakpoints tv(1),...,tv(N+1). The coordinates of the points in each of the patches are given by the general bicubic polynomials (given here for the (i, j) Patch).

$$
\begin{aligned}
X(u,v) &= AX(i,j)+BX(i,j)*s+CX(i,j)*s^2+DX(i,j)*s^3 \\
&+ EX(i,j)*t+FX(i,j)*t*s+GX(i,j)*t*s^2+HX(i,j)*t*s^3 \\
&+ KX(i,j)*t^2+LX(i,j)*t^2*s+MX(i,j)*t^2*s^2+NX(i,j)*t^2*s^3 \\
&+ PX(i,j)*t^3+QX(i,j)*t^3*s+RX(i,j)*t^3*s^2+SX(i,j)*t^3s^3
\end{aligned}
$$

$$Y(u,v) = \ldots$$

$$Z(u,v) = \ldots$$

where

$$tu(j) \leq u \leq tu(j+1), \qquad j=1,M$$

$$s=u-tu(j)$$

and

$$tv(i) \leq v \leq tv(i), \qquad i=1,N$$

$$t=v-tv(i).$$

To provide edge values and derivatives without evaluating polynomials, an additional dummy row and column of patches is included in the entity.

An example of the bicubic surface is shown in Fig. 3.1-8.

Directory Data

ENTITY NUMBER : 114

Parameter Data

Parameter	Value	Format	Comment
1	STYPE	Integer	Spline Type (1=Linear 2=Quadratic 3=Cubic 4=Wilson-Fowler 5=Modified Wilson-Fowler)
2	PTYPE	Integer	Patch Type (1=Cartesian Product 0=Not Necessarily)
3	M	Integer	Number of u segments
4	N	Integer	Number of v segments
5	tu(1)	Floating Point	Breakpoints in u:
.	.		u values of grid
.	.		lines
.	.		
5+M	tu(M+1)		

Parameter	Value	Format	Comment
6+M	tv(1)	Floating Point	Breakpoints in v values of gri lines
	.		
	.		
	.		
6+M+N	tv(N+1)		
7+M+N	AX(1,1)	Floating Point	X Coefficient: (1,1) Patch
.	.		
.	.		
.	.		
22+M+N	SX(1,1)		
23+M+N	AY(1,1)		Y Coefficient: (1,1) Patch
.	.		
.	.		
.	.		
38+M+N	SY(1,1)		
39+M+N	AZ(1,1)		Z Coefficient: (1,1) Patch
.	.		
.	.		
.	.		
54+M+N	SZ(1,1)		
.	.		Coefficients c
.	.		
.	.		(1,2) Patch
			.
			.
			.
			(1,N+1)
			(2,1) Patch
			.
			.
			.

Parameter	Value	Format	Comment
			(2, N+1)
			.
			.
			.
			(M+1,1) Patch
			.
			.
			.
			(M+1,N+1)

TOTAL ENTRIES = 6+M+N+48*(M+1)*(N+1)=TE

TE+1	NA	Integer	Number of associated entities
TE+2	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
TE+1+NA	DE	Integer	
TE+2+NA	MA	Integer	Number of associated properties
TE+3+NA	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
TE+2+NA+MA	DE	Integer	

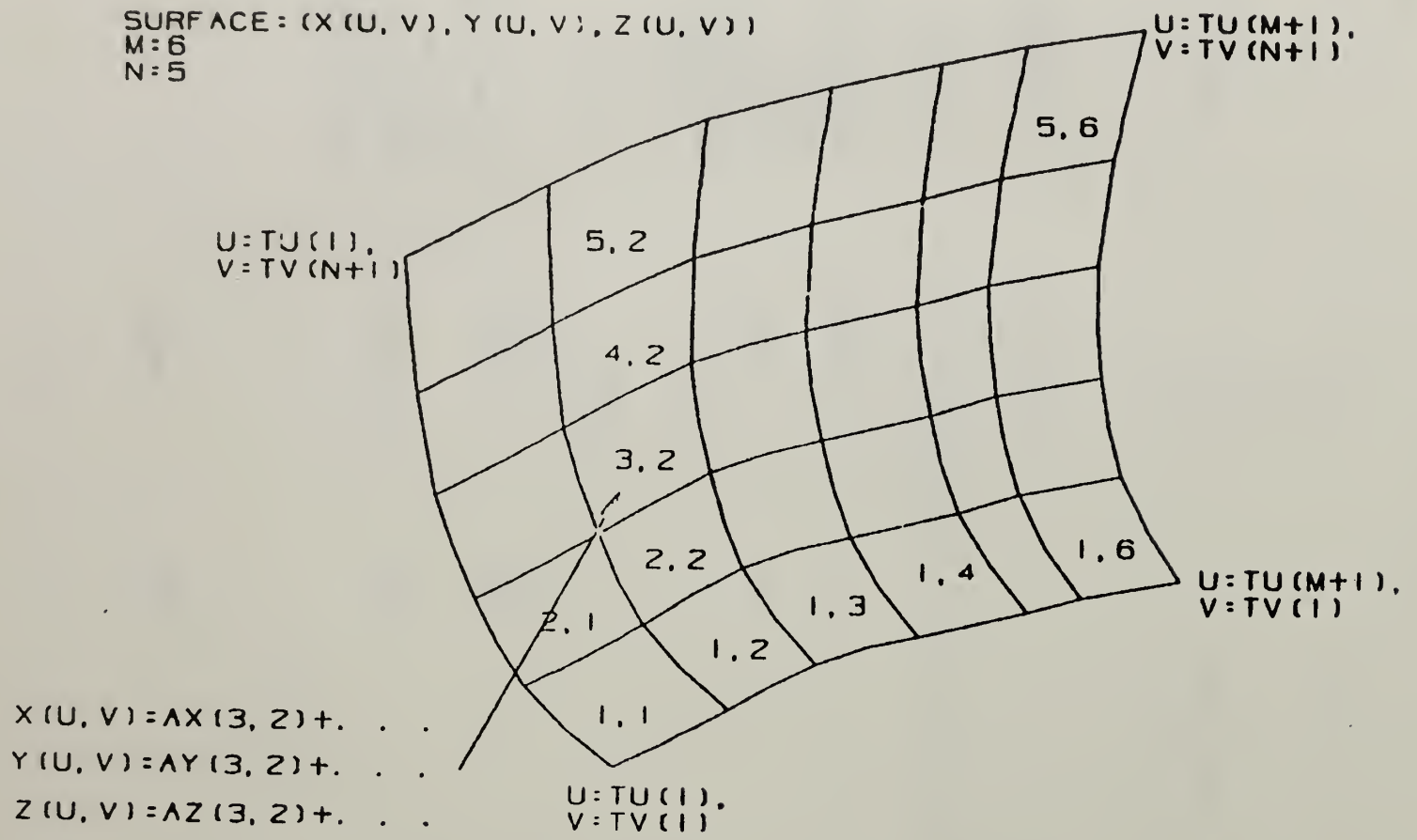


FIG. 3.1-8 EXAMPLE OF PARAMETRIC BICUBIC SPLINE SURFACE

POINT

A point is defined by its coordinates in three dimensions. Examples of the point entity are shown in Fig. 3.1-9.

Directory Data

ENTITY NUMBER : 116

PARAMETER DATA

Parameter	Value	Format	Comment
1	x	Floating Point	Coordinate
2	y	Floating Point	of
3	z	Floating Point	Point
4	PTR	Integer	Pointer to DE of subfigure instance specifying the display symbol. If 0, no display symbol specified.
5 N		Integer	Number of associated entities
6 DE		Integer	Pointers to DEs of associated entities
. .	.		
. .	.		
. .	.		
5+N DE		Integer	
6+N M		Integer	Number of associated properties
7+N DE		Integer	Pointers to DEs of associated properties
. .	.		
. .	.		
. .	.		
6+N+M		DE Integer	

EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.1-9 EXAMPLES OF POINTS

RULED SURFACE

A ruled surface is formed by moving a line connecting points of equal relative arc length on two curves from a start point to an end point on the curves. The curves may be points, lines, circles, conics, or parametric splines (both 2D and 3D).

If the two curves are expressed as the parametric functions of arc length

$(C1_X(s), C1_Y(s), C1_Z(s))$ and $(C2_X(t), C2_Y(t), C2_Z(t))$, then

the coordinates of the points on the ruled surface can be written as

$$X(u,v) = (1-v)*C1_X(s)+v*C2_X(t)$$
$$Y(u,v) = (1-v)*C1_Y(s)+v*C2_Y(t)$$
$$Z(u,v) = (1-v)*C2_Z(s)+v*C2_Z(t)$$

where

$0 \leq u \leq 1$, $0 \leq v \leq 1$

s = u * arc length (C1)

t = u * arc length (C2).

If DIRFLG=1, then the curves are joined first to last, last to first and t is given by

t = (1-u) * arc length (C2).

If DEVFLG=1, then the surface is a developable surface; if DEVFLG=0, the surface may or may not be a developable surface.

An example is shown in Fig. 3.1-10. Additional examples are shown in Fig. 3.1-11.

Directory Data

ENTITY NUMBER : 118

Parameter Data

Parameter	Value	Format	Comment
1	DE1	Integer	Pointer to curve
2	DE2	Integer	Pointer second curv
3	DIRFLG	Integer	Direction F (0=join firs first, last last 1=joint firs last, last first)
4	DEVFLG	Integer	Developable surface flag (1=Develop 0=Possibly
5	N	Integer	Number associated entities
6	DE	Integer	Pointers to of associ entities
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Integer	

Parameter	Value	Format	Comment
6+N	M	Integer	Number of associated properties
7+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Integer	

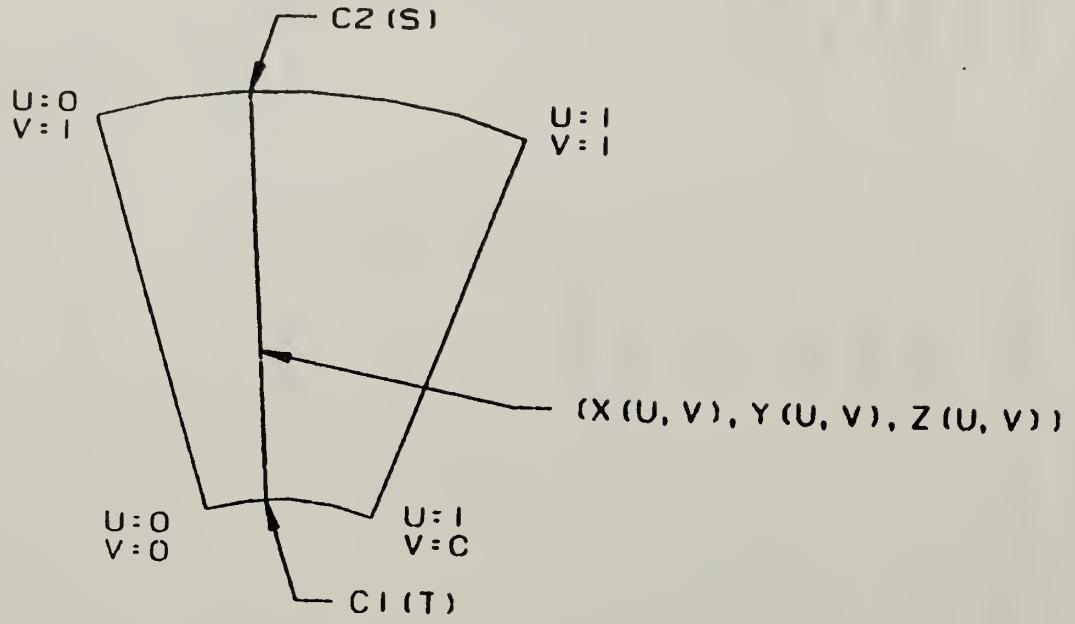
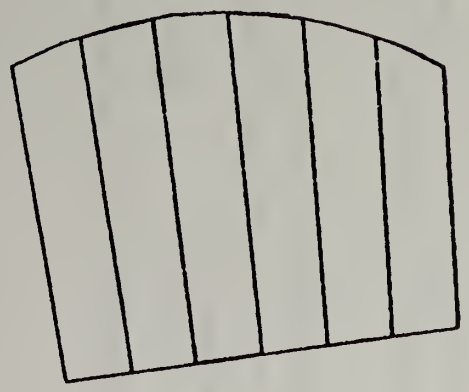
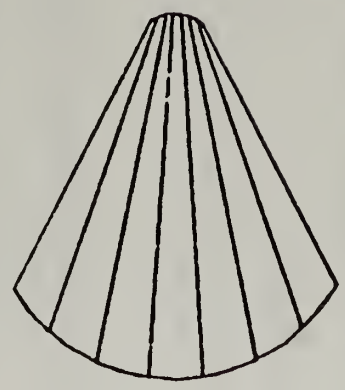


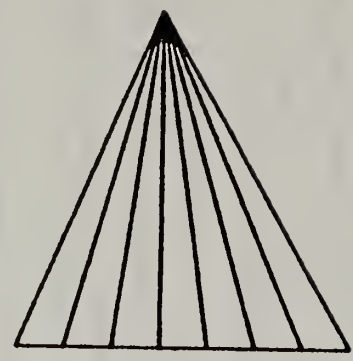
FIG. 3.1-10 EXAMPLES OF RULED SURFACE



EXAMPLE 1



EXAMPLE 2



EXAMPLE 3

FIG. 3.1-11 EXAMPLES OF RULED SURFACES

SURFACE OF REVOLUTION

A surface of revolution is defined by a directrix (which must be a line), a generatrix, and start and end rotation angles. The surface is created by rotating the generatrix about the directrix thru the start and end angles. The angles of rotation are measured via the right hand screw rule. There is no transformation matrix associated with a surface of revolution. A matrix may be associated with the generatrix, which may be a conic, line, arc (circle), parametric spline, or composite entity.

Examples of surface of revolution entities are shown in Fig. 3.1-13.

The start and end angles of the surface can be explained by geometric construction. Refer to Fig. 3.1-12 and the following:

(1) Select a point on the generatrix which does not lie on the directrix; label the point P1.

(2) Construct a line through P1 such that it is perpendicular to the directrix; label this line L1.

(3) Construct a plane PN1 containing L1 and perpendicular to the directrix.

(4) All rotations are applied in the plane PN1 about the directrix according to the right hand rule.

(5) Rotate the line L1 and the point selected from the curve the number of radians indicated in the start angle resulting in $L1_{SA}$. The location is labeled LOC1.

(6) Rotate the line L1 and the point selected from the curve an additional number of radians given by the end angle minus the start angle resulting in $L1_{SE}$. The second location of the point is labelled LOC2.

(7) The resulting surface is that generated by rotating the curve from LOC1 to LOC2.

Directory Data

ENTITY NUMBER : 120

Parameter Data

Parameter	Value	Format	Comment
1	DE	Integer	Pointer to a line DE (directrix)
2	DE	Integer	DE pointer to generatrix
3	SA	Floating point	Start angle in radians
4	EA	Floating point	End angle in radians
5	N	Integer	Number of associated entities
6	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Integer	
6+N	M	Integer	Number of associated properties
7+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Integer	

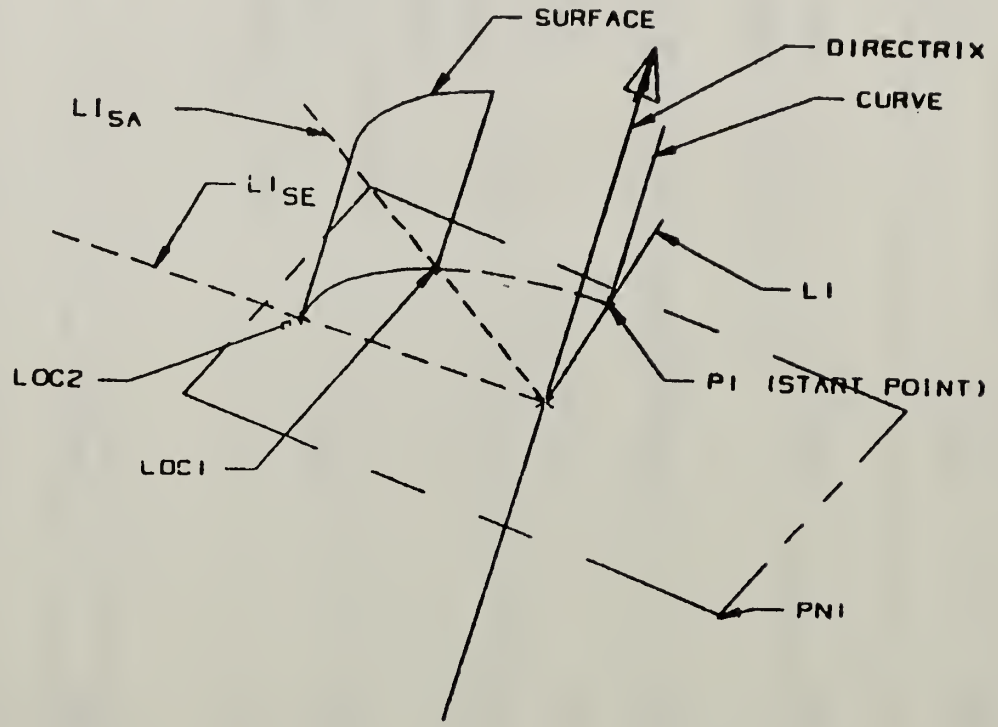
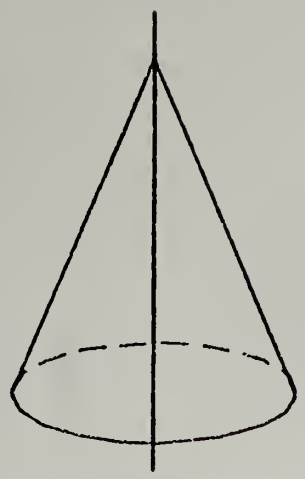
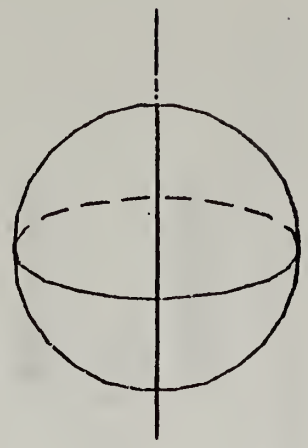


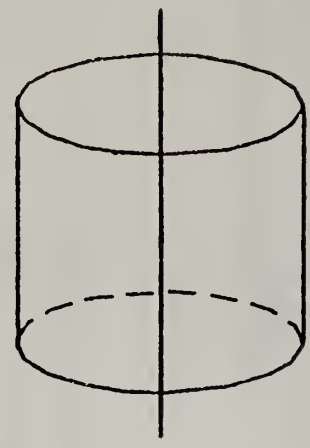
FIG. 3.1-12 SURFACE OF REVOLUTION START AND END ANGLES



EXAMPLE 1



EXAMPLE 2



EXAMPLE 3

FIG. 3.1-13 EXAMPLES OF SURFACE OF REVOLUTION ENTITIES

TABULATED CYLINDER

A tabulated cylinder is formed by moving the generatrix (a line) parallel to itself al the directrix which may be a line, circle, conic, or parametric spline.

The points on the surface are given by

$$X(u,v) = CX(u)+v*(LX-CX(0))$$
$$Y(u,v) = CY(u)+v*(LY-CY(0))$$
$$Z(u,v) = CZ(u)+v*(LZ-CZ(0))$$

where $0 \leq u \leq 1, \quad 0 \leq v \leq 1$

and CX, CY, CZ represents the X, Y, Z component along the direct curve.

An example of the tabulated cylinder is shown in Fig. 3.1-14.

Directory Data

ENTITY NUMBER : 122
ATTRIBUTES REQUIRED : MTX

Parameter Data

Parameter	Value	Format	Comment
1	DE	Integer	Pointer to direct curve
2	LX	Floating Point	Coordinates of
3	LY	Floating Point	endpoint of the g
4	LZ	Floating Point	erating line. other end point is first point on directrix.
5	N	Integer	Number of associa entities
6	DE	Integer	Pointers to DEs associated entities.
.	.	.	
.	.	.	
.	.	.	

Parameter	Value	Format	Comment
5+N	DE	Integer	
6+N	M	Integer	Number of associated properties
7+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Integer	

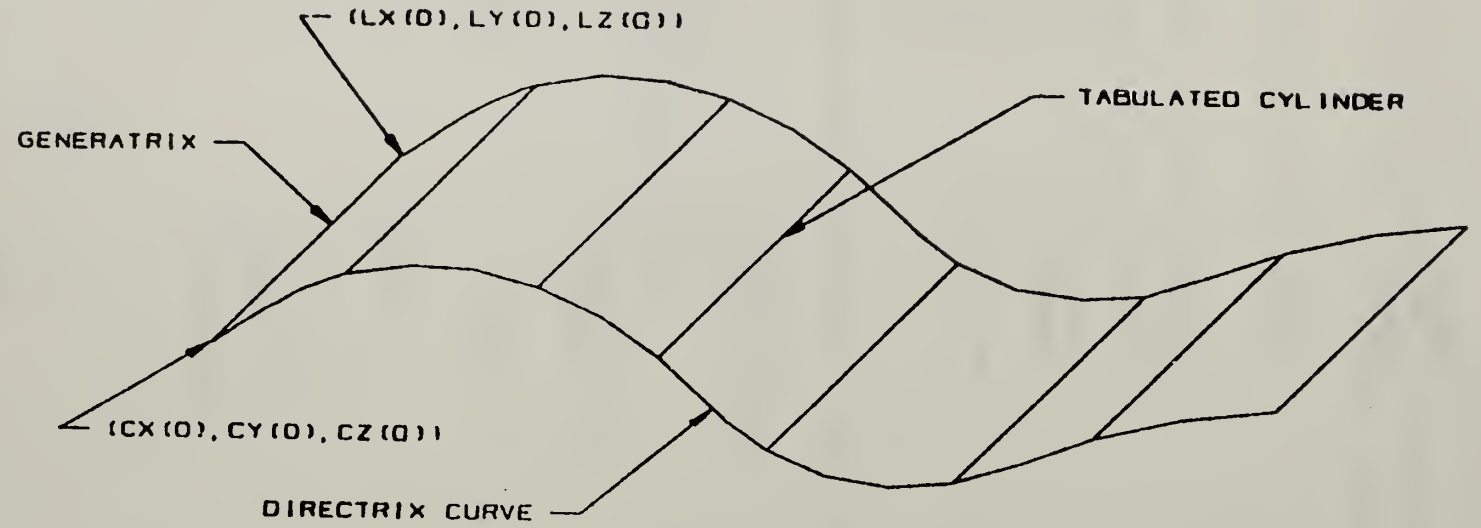


FIG. 3.1-14 EXAMPLES OF A TABULATED CYLINDER

TRANSFORMATION MATRIX

A twelve cell entity which contains 3x3 rotation matrix R and a one by three translation matrix T. The transformation moves the associated entity from its definition space to model space, by first applying the rotation, then translation.

Directory Data

ENTITY NUMBER : 124

Parameter Data

Parameter	Value	Format	Comment
1	R11	Floating Point	Top Row
2	R12	Floating Point	
3	R13	Floating Point	
4	T1	Floating Point	
5	R21	Floating Point	Second Row
6	R22	Floating Point	
7	R23	Floating Point	
8	T2	Floating Point	
9	R31	Floating Point	Third Row
10	R32	Floating Point	
11	R33	Floating Point	
12	T3	Floating Point	

Parameter	Value	Format	Comment
13	N	Integer	Number of associa entities
14	DE	Integer	Pointers to DEs associated entities
.	.	.	
.	.	.	
.	.	.	
13+N	DE	Integer	
14+N	M	Integer	Number of associa properties
15+N	DE	Integer	Pointers to DEs associated properties
.	.	.	
.	.	.	
.	.	.	
14+N+M	DE	Integer	

3.2 Drafting Entities

Many drafting entities are constructed by using other entities. For example the linear dimension entity will contain a pointer to a witness line directory entry (a form of copious data), two pointers to leader (arrow) directory entries, and one pointer to a general note directory entry.

All drafting entities are defined in a definition space (XT, YT). This (XT, YT) space may be identical to the (XT, YT) space of a drawing entity (No. 404). Alternatively, a reference may be made to a transformation matrix. The transformation matrix rotates the drafting entity into model space. Subordinate entities to a drafting entity may have different ZT displacements. For example within the linear dimension, a different ZT value may be found in each of: general note, leader, and witness lines (which are pointed to in the linear dimension parameter data). An example showing the use of ZT displacement (DEPTH) is shown in Fig. 3.2-1.

The following entities are defined in this section:

Entity Name	Entity Number
Angular Dimension	202
Centerline	106
Diameter Dimension	206
Flag Note	208
General Label	210
General Note	212
Leader (Arrow)	214
Linear Dimension	216
Ordinate Dimension	218
Point Dimension	220
Radius Dimension	222
Section	106
Witness Line	106

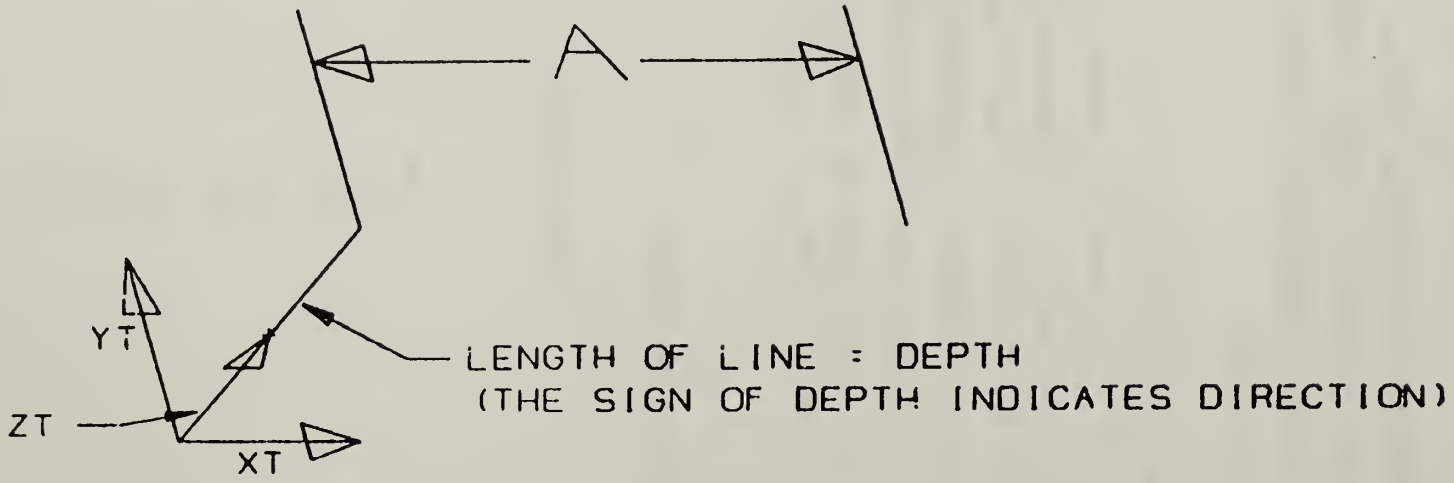


FIG. 3.2-1 CONSTRUCTION OF ZT DEPTH OF DRAFTING ENTITIES

ANGULAR DIMENSION

An angular dimension consists of a general note, zero to two witness lines, (if two witness lines are needed, both are contained in the copious data of the witness line entity), zero to two arrows, and an angle vertex point. The first segment of each leader is an arc with its center at the vertex point. An example of this arc construction is shown in Fig. 3.2-2. Remaining leader segments, if any, are straight lines. See Leader (Arrow) for details.

Refer to Fig. 3.2-3 for examples of angular dimensions.

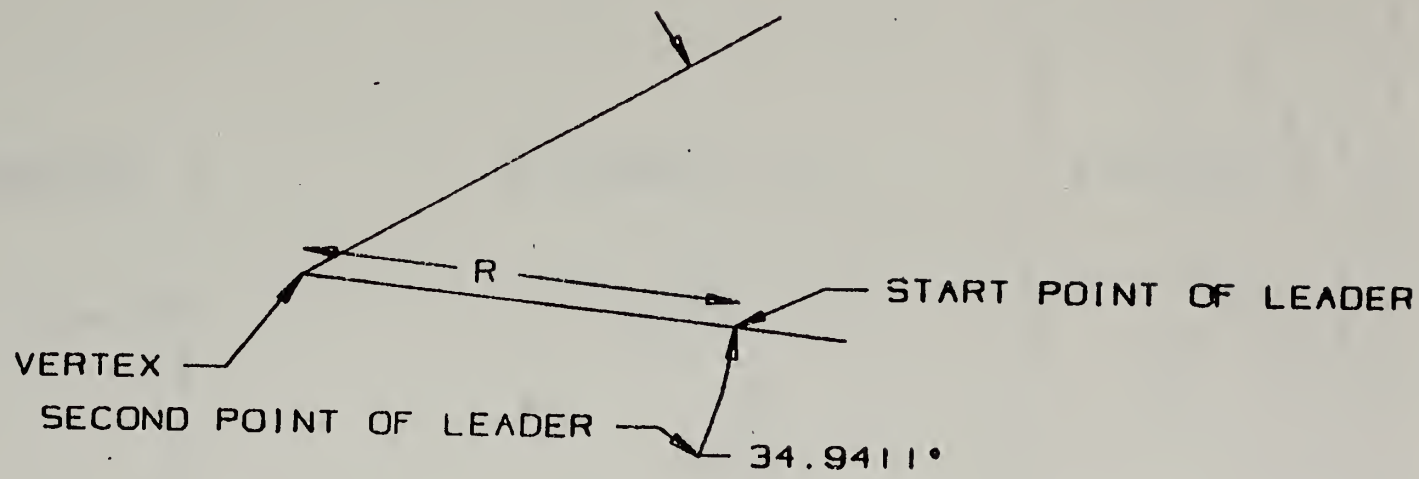
Directory Data

ENTITY NUMBER : 202

Parameter Data

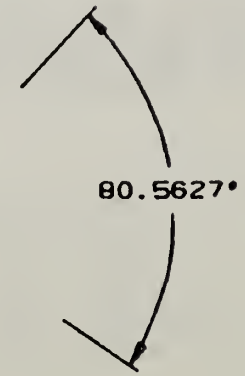
Parameter	Vaiue	Format	Comment
1	GN	Integer	Pointer to general note DE
2	W1	Integer	Pointer to witness line DE or 0
3	XT	Floating Point	Coordinates of vertex point
4	YT	Floating Point	
5	R	Floating Point	Radius of leader arcs
6	A1	Integer	Pointer to 1st leader DE or 0

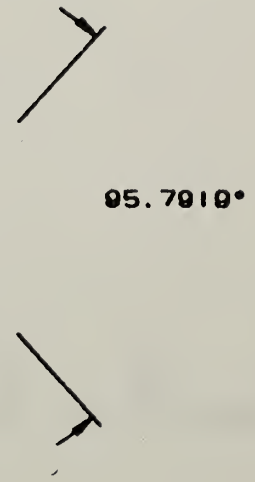
Parameter	Value	Format	Comment
7	A2	Integer	Pointer to 2nd le DE or 0
8	N	Integer	Number of assoc. entities
9	DE	Integer	Pointers to DE associated entiti
.	.	.	
.	.	.	
.	.	.	
8+N	DE	Integer	
9+N	M	Integer	Number of assoc properties
10+N	DE	Integer	Pointers to DE associated prope
.	.	.	
.	.	.	
.	.	.	
9+N+M	DE	Integer	



THE RADIUS OF THE ARC IN THE LEADER MUST BE CALCULATED BETWEEN THE VERTEX POINT AND THE START POINT OF THE LEADER

FIG. 3.2-2 ANGULAR DIMENION: CONSTRUCTION OF ARCS IN THE ASSOCIATED LEADERS





EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.2-3 EXAMPLES OF ANGULAR DIMENSION

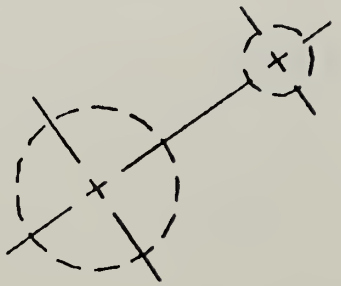
CENTERLINE

The Centerline entities are stored as a form of copious data. The associated matrix transforms the XT-YT plane of the centerline into model space. The coordinates of the centerline points describe the centerline display symbol. The display symbol is described by line segments where each line is from

(X_n, Y_n, Z_n), to $(X_{n+1}, Y_{n+1}, Z_{n+1})$ where
$n = 1,3,5,\ldots,N-1$

Examples of the centerline entity are shown in Fig. 3.2-4.

EXAMPLE 1



EXAMPLE 2

EXAMPLE 3

FIG. 3.2-4 EXAMPLES OF CENTERLINE ENTITIES

DIAMETER DIMENSION

A diameter dimension consists of a general note, one or two leaders, and an arc center point. If a second leader does not exist its pointer value will be 0. Refer to Fig. 3.2-5 for examples of the diameter Dimension entity. The arc center coordinates are used for positioning the diameter dimension line relative to the arc being dimensioned.

Directory Data

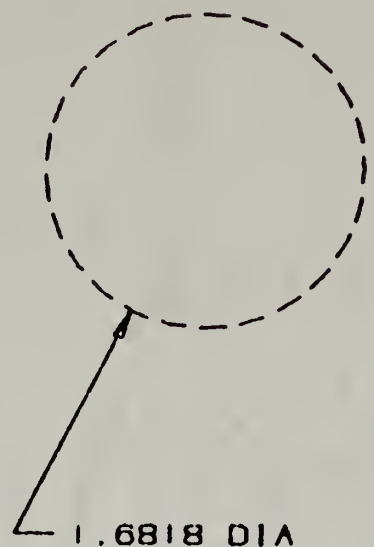
ENTITY NUMBER : 206

Parameter Data

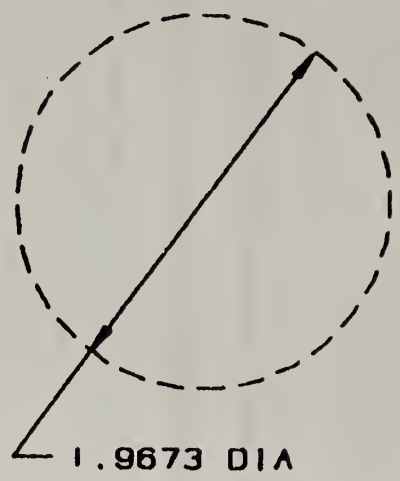
Parameter	Value	Format	Comment
1	NE	Integer	Pointer to general note DE
2	A1	Integer	Pointer to first leader DE
3	A2	Integer	Pointer to second leader DE
4	XT	Floating Point	Arc center coordinates
5	YT	Floating Point	
6	N	Integer	Number of associated entities

Parameter	Value	Format	Comment
7	DE	Integer	Pointers to DEs associated entitie
.	.	.	
.	.	.	
.	.	.	
6+N	DE	Integer	
7+N	M	Integer	Number of associ properties
8+N	DE	Integer	Pointers to DEs associated proper
.	.	.	
.	.	.	
.	.	.	
7+N+M	DE	Integer	

EXAMPLE 1



EXAMPLE 2



EXAMPLE 3

FIG. 3.2-5 EXAMPLES OF DIAMETER DIMENSIONS

FLAG NOTE

A flag note is label information formatted as shown in Figure 3.2-6. The rotation angle overrides the general note rotation angle and placement. Additional examples of the flag note entity are shown in Fig. 3.2-7.

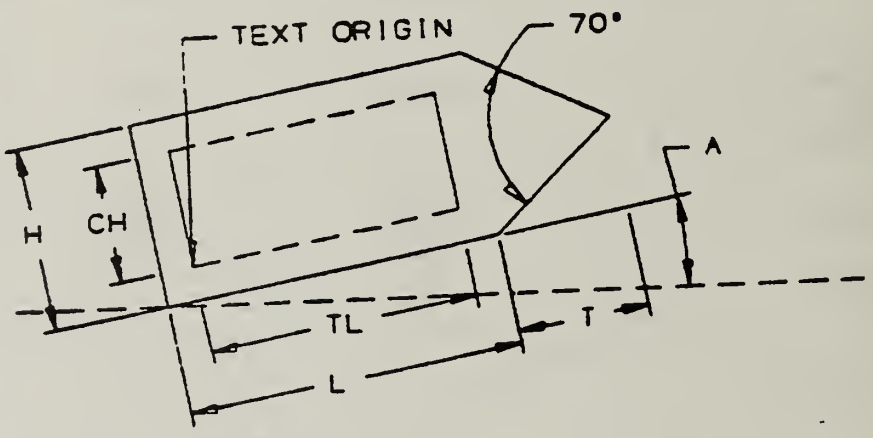


Fig. 3.2-6

The following specifications apply to Fig. 3.2-6.

Variables:

H	=	Height	CH	=	Character Height
L	=	Length	NC	=	Number of Characters (in general note)
TL	=	Text Length			
T	=	Tip Length	A	=	Rotation angle in radians

Formulas:

TL = (.8)(CH)(NC)+(.4)(CH)(NC-1)

H = (2)(CH)

L = (TL)+(.4)(CH)

T = (.5)(H)/TAN35°

Restrictions:

H shall never be less than .3 in.

L shall never be less than .6 in.

T shall never be less than .214 in.

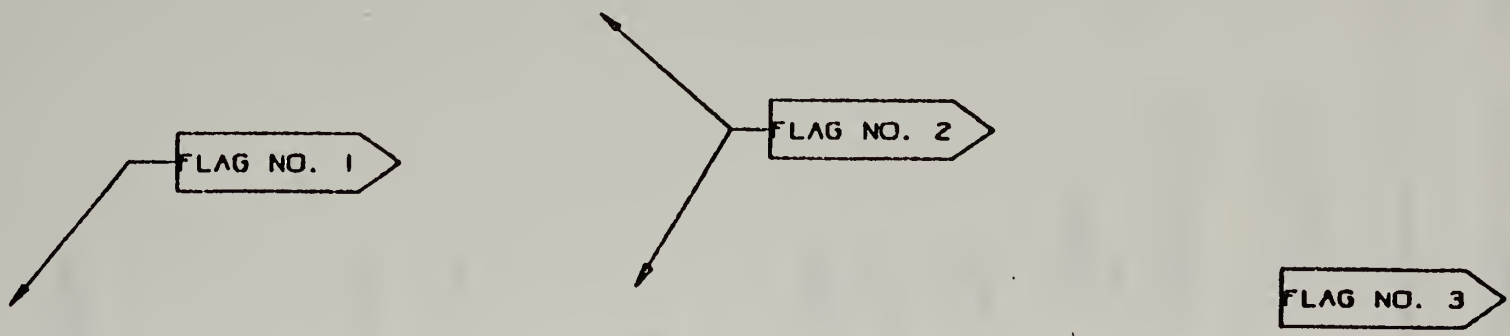
Directory Data

ENTITY NUMBER : 208

Parameter Data

Parameter	Value	Format	Comment
1	XT	Floating Point	Lower left corner
2	YT	Floating Point	Coordinate
3	ZT	Floating Point	
4	A	Floating Point	Rotation angle in radians
5	DENOTE	Integer	Pointer to general note DE
6	N	Integer	Number of arrows (leaders)
7	DE	Integer	DE pointers to associated leaders DEs
.			
.			
.			
6+N			

Parameter	Value	Format	Comment
7+N	NA	Integer	Number of associa entities
8+N	DE	Integer	Pointers to DEs associated entitie
.	.	.	
.	.	.	
.	.	.	
7+N+NA	DE	Integer	
8+N+NA	M	Integer	Number of associa properties
9+N+NA	DE	Integer	Pointers to DEs associated propert
.	.	.	
.	.	.	
.	.	.	
6+N+NA+M	DE	Integer	



EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.2-7 EXAMPLES OF FLAG NOTES

GENERAL LABEL

A general label consists of a general note with one or more associated leaders.

Examples of general label entities are shown in Fig. 3.2-8.

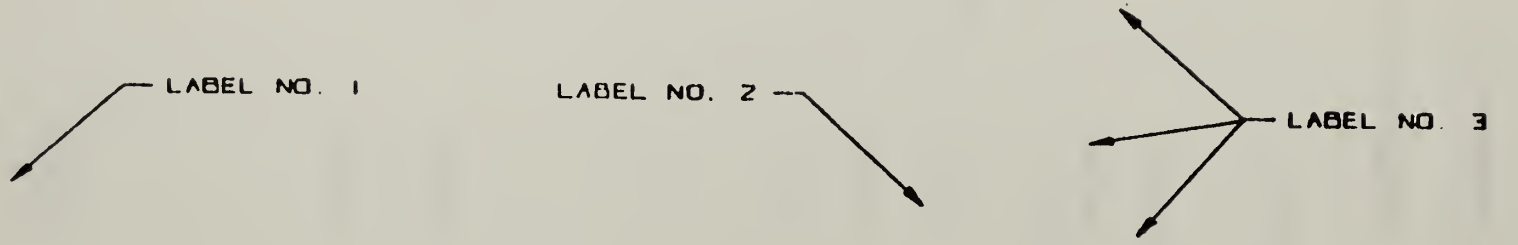
Directory Data

ENTITY NUMBER : 210

Parameter Data

Parameter	Value	Format	Comment
1	DENOTE	Integer	Pointer to assoc general note
2	N	Integer	Number of leade
3	DE	Integer	Pointer to assoc leaders DEs
.	.	.	
.	.	.	
.	.	.	
N+2			
N+3	NA	Integer	Number of assoc entities
N+4	DE	Integer	Pointers to DE associated entiti
.	.	.	
.	.	.	
.	.	.	
N+3+NA	DE	Integer	

Parameter	Value	Format	Comment
N+4+NA	M	Integer	Number of associated properties
N+5+NA	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
N+4+NA+M	DE	Integer	



EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.2-8 EXAMPLES OF GENERAL LABELS

GENERAL NOTE

A general note consists of text, a starting point, text size, and angle of rotation of th text. If a transformation matrix is specified, it transforms the note from the XT-Y plane to model space. Examples of general notes are shown in Fig. 3.2-9. The F value indicates the font characteristic and is an integer between 0 and 6. This integ will be obtained on a system-dependent basis using the information in Figures 3.2-1 and 3.2-11. If the FC number is not sufficient to describe the font, a text for definition entity may be used to define the font. If a text font definition is being use the negative of the pointer value for the DE of the text font definition entity is place in the FC parameter. The use of the values WT, HT, SL, A, and text start point, a shown in Fig. 3.2-12.

Directory Data

ENTITY NUMBER : 212

Parameter Data

Parameter	Value	Format	Comment
1	NS	Integer	Number of text strings General Note
2	NC	Integer	Number of characters the text
3	WT	Floating Point	Character string length
4	HT	Floating Point	Character height
5	FC	Integer	Font characteristic
6	SL	Floating Point	Slant angle of text radians
7	A	Floating Point	Rotation angle in radians
8	M	Integer	Mirror flag (0-no mirror, text mirrored)
9	VH	Integer	Vertical/Horizontal orientation flag (0-te horizontal, 1-text vertical

Parameter	Value	Format	Comment
10	XT	Floating Point	Text start point
11	YT	Floating Point	
12	ZT	Floating Point	Z Depth from XT, YT plane
13	TEXT	Character	First text string
14	NC2	Integer	Number of characters in second text string
.			
.			
.			
1+NS*12	TEXT	Character	last text string
2+NS*12	N	Integer	Number of associated entities
3+NS*12	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
2+NS*12	DE	Integer	
3+NS*12	M	Integer	Number of associated properties
4+NS*12	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
3+NS*12	DE	Integer	

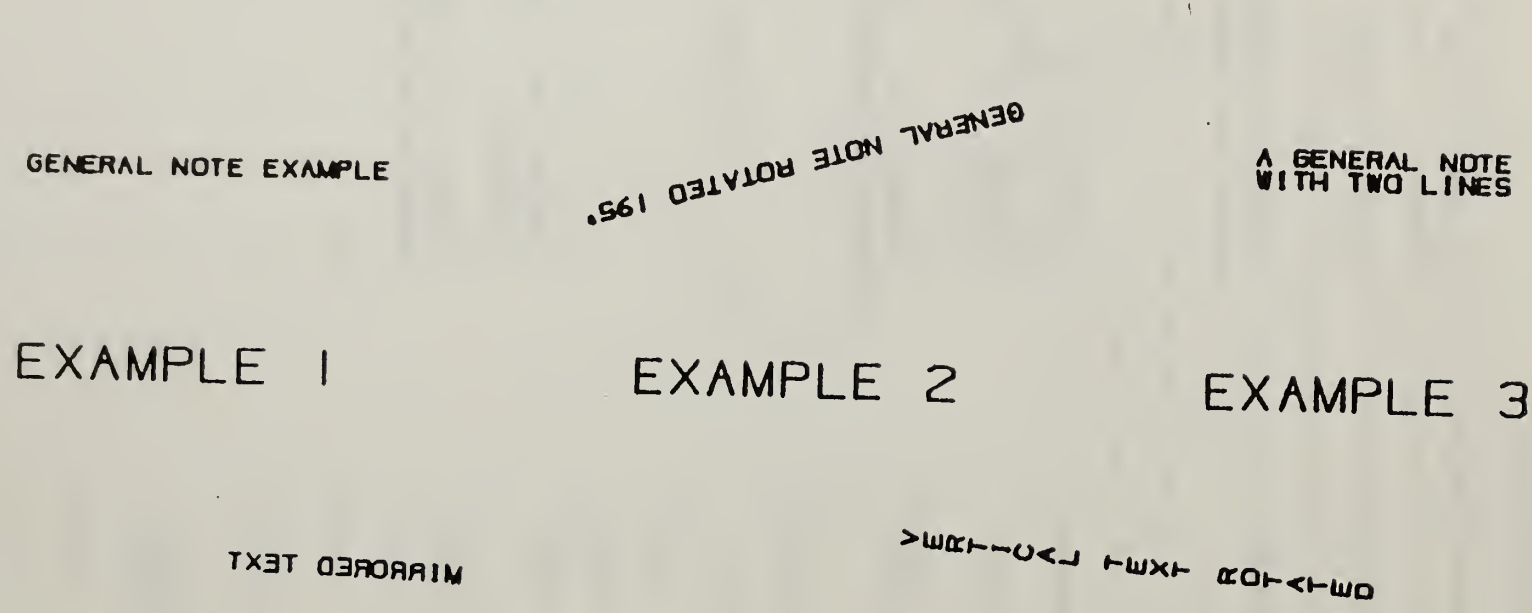


FIG. 3.2-9 EXAMPLES OF GENERAL NOTES

FONT CODE (FC)	FONT APPEARANCE
1	ABCDEFGHIJKLMNOPQRSTUVWXYZ12345678901 "#%&'()×+,-./::;?
2	ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890%()*+.-./::
3	ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890()+,-./:;
4	ABCDEFGHIJKLMNOPQRSTUVWXYZ12345678901"#$%&'()*+,-./:;?
5	ABCDEFGHIJKLMNOPQRSTUVWXYZ12345678901"#$%&'()*+,-./:
6	ABCDEFGHIJKLMNOPQRSTUVWXYZ12345678901"#$%&'()*+,-./::?

FIG. 3.2-10 FONT CODES FOR FONTS OF ASCII CHARACTERS

0	⧖	27	ω	56	.	105	E	134	\	163	Ⓢ
1	÷	30	λ	57	/	106	F	135	]	164	⊡
2	≤	31	α	60	0	107	G	136	^	165	⏀
3	≥	32	δ	61	1	110	H	137	_	166	⚠
4	△	33	μ	62	2	111	I	140	`	167	⏁
5	√	34	π	63	3	112	J	141	a	170	⏃
6	X	35	¯	64	4	113	K	142	⊞	171	⧗
7	≡	36	±	65	5	114	L	143	▱	172	Y
10	≠	37	°	66	6	115	M	144	⌓	173	{
11	∫	40	SP	67	7	116	N	145	◯	174	|
12	⊃	41	!	70	8	117	O	146	//	175	}
13	∨	42	"	71	9	120	P	147	⌀	176	~
14	∧	43	#	72	:	121	Q	150	⬈	177	⏄
15	≈	44	$	73	;	122	R	151	⩵		
16	Σ	45	%	74	<	123	S	152	⊕		
17	↑	46	&	75	=	124	T	153	⌒		
20	↓	47	'	76	>	125	U	154	⊥		
21	→	50	(	77	?	126	V	155	Ⓜ		
22	←	51	)	100	@	127	W	156	∅		
23	φ	52	*	101	A	130	X	157	○		
24	θ	53	+	102	B	131	Y	160	Ⓟ		
25	τ	54	,	103	C	132	Z	161	℄	FONT CODE =	
26	ψ	55	—	104	D	133	[	162	◎	ASCII CODE CHARACTER N	

Figure 3.2-11.

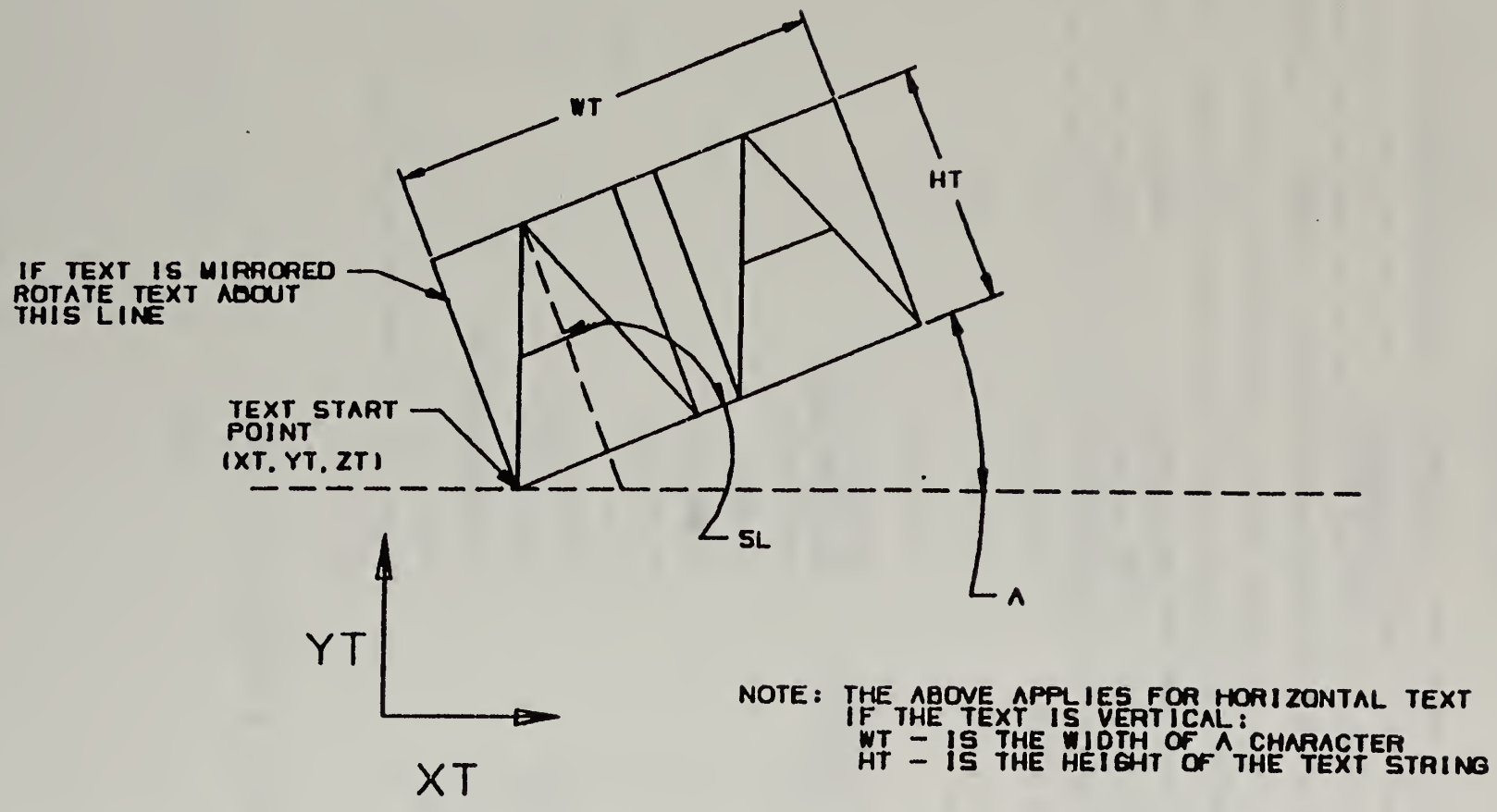


FIG. 3.2-12 GENERAL NOTE TEXT CONSTRUCTION

LEADER (ARROW)

A leader consists of one or more line segments except when the leader is part of an Angular dimension (see Angular dimension). The first segment begins with an arrowhead. There shall be ten different arrowheads and the selection shall be made by assigning values to FORM (see table below). Remaining segments successively link to a presumed text item. The leader entity includes parameters to define the size and shape of the arrowhead and the end points of each segment of the leader. Individual segments are assumed to extend from the end point of its predecessor in the segment list to its defined end point. Points are defined in an XT-YT plane which is rotated into the model space by the associated transformation matrix. Examples of leaders are shown in Fig. 3.2-13.

FORM Definition:

FORM NUMBER	Meaning Arrowhead Type
1	Open Triangle
2	Triangle
3	Triangle (filled in)
4	No arrowhead
5	Circle
6	Circle (filled in)
7	Rectangle
8	Rectangle (filled in)
9	Slash
10	Integral sign centered at the end point of the first segment

Examples of each FORM are shown in Fig. 3.2-14.

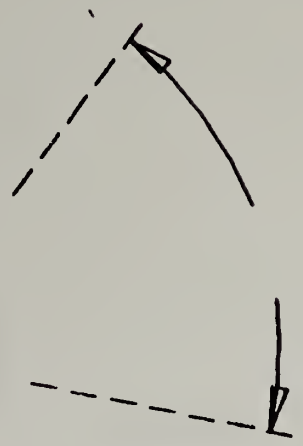
Directory Data

ENTITY NUMBER: 214
ATTRIBUTES REQUIRED: FORM

Parameter Data

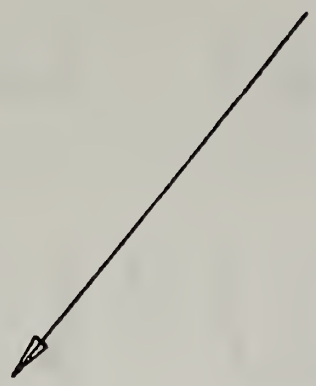
Parameter	Value	Format	Comment
1	N	Integer	Number of Segments
2	AD1	Floating Point	Arrowhead Height
3	AD2	Floating Point	Arrowhead Width
4	ZT	Floating Point	Z Depth
5	XH	Floating Point	Arrowhead Coordinates
6	YH	Floating Point	
7		Floating Point	Segment tail
.			coordinate pairs
.			
.			
6+2N			
7+2N	NA	Integer	Number of associated entities
8+2N	DE	Integer	Pointer to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
7+2N+NA	DE	Integer	

Parameter	Value	Format	Comment
8+2N+NA	M	Integer	Number of associ properties
9+2N+NA	DE	Integer	Pointers to DEs associated properti
.	.	.	
.	.	.	
.	.	.	
8+2N+NA+M	DE	Integer	



EXAMPLE 1

(ANGULAR DIMENSION)



EXAMPLE 2

EXAMPLE 3

FIG. 3.2-13 EXAMPLES OF LEADERS

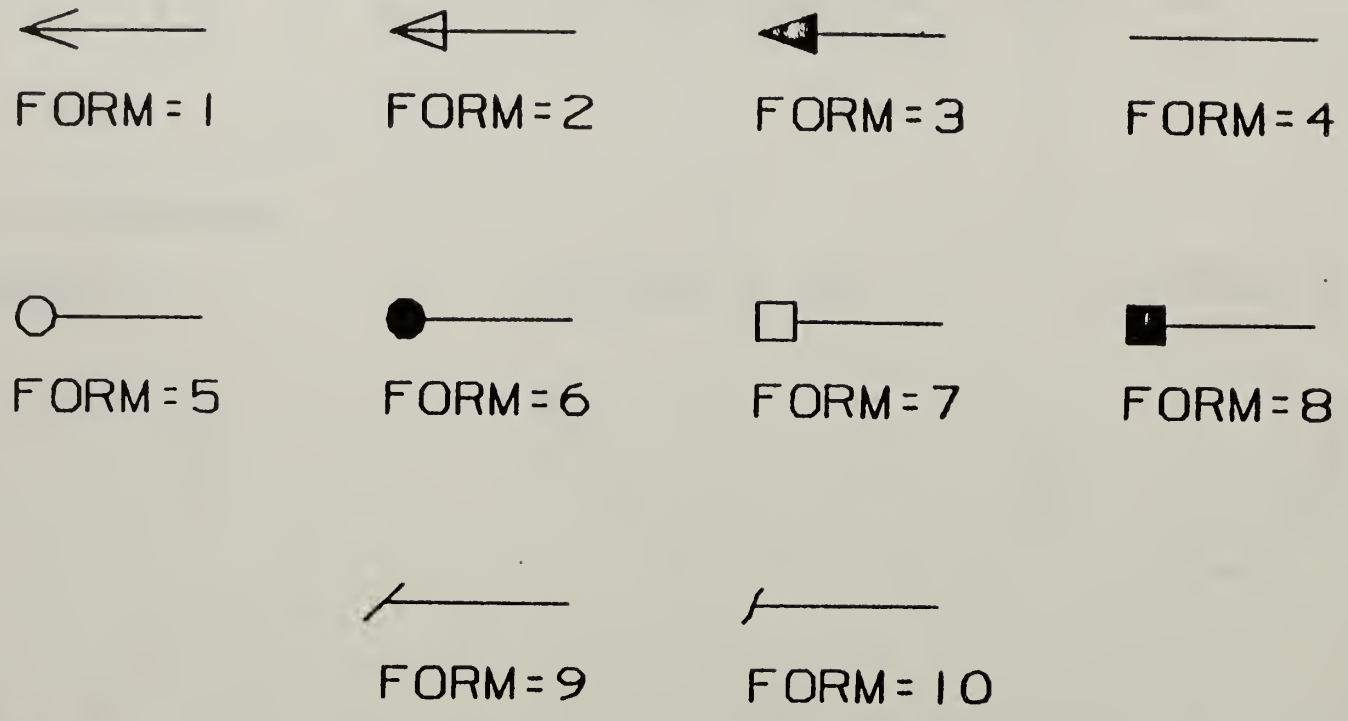


FIG. 3.2-14 ARROWHEAD DEFINITIONS

LINEAR DIMENSION

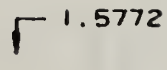
A linear dimension consists of a general note, two leaders, and zero to two witness lines. Refer to Fig. 3.2-15 for examples of linear dimensions. If two witness lines are required, both are stored in the copious data of the witness line.

Directory Data

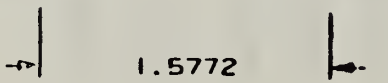
ENTITY NUMBER : 216

Parameter Data

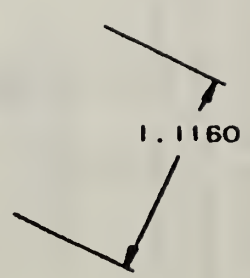
Parameter	Value	Format	Comment
1	DENOTE	Integer	Pointer to General note DE
2	DEARRW1	Integer	Pointer to first leader DE
3	DEARRW2	Integer	Pointer to second leader DE
4	DEWIT1	Integer	Pointer to witness line DE, 0 if not defined
5	N	Integer	Number of associated entities
6	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Integer	
6+N	M	Integer	Number of associated properties
7+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Integer	



EXAMPLE 1



EXAMPLE 2



EXAMPLE 3

FIG. 3.2-15 EXAMPLES OF LINEAR DIMENSIONS

ORDINATE DIMENSION

The ordinate dimension is used to indicate dimensions from a common base line. Dimensioning is only permitted along the XT or YT axis.

An ordinate dimension consists of a general note and a witness line or leader. The values stored are pointers to the directory entry for the associated note and witness line. Examples of ordinate dimensions are shown in Fig. 3.2-16.

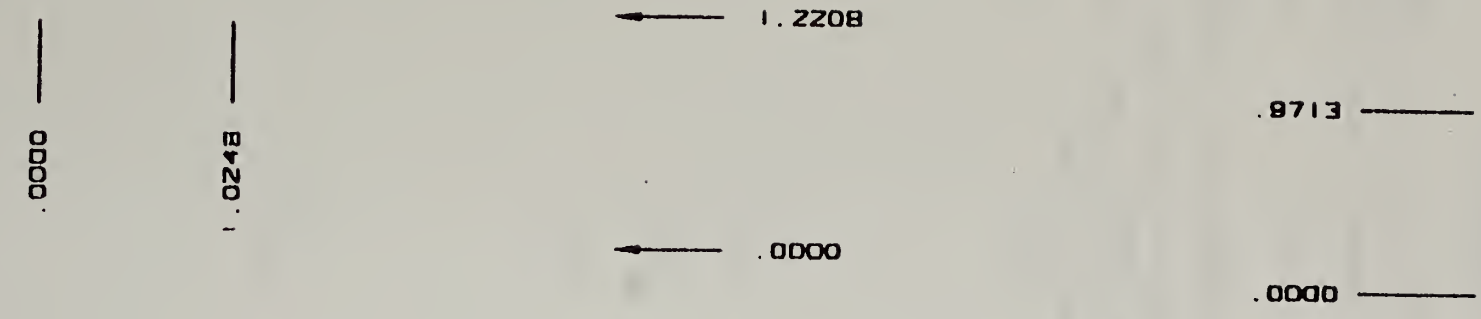
Directory Data

ENTITY NUMBER : 218

Parameter Data

Parameter	Value	Format	Comment
1	NOTE	Integer	Pointer to General note DE
2	WIT	Integer	Pointer to Witness line DE or leader DE
3	N	Integer	Number of associated entities
4	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
3+N	DE	Integer	
4+N	M	Integer	Number of associated properties

Parameter	Value	Format	Comment
5+N	DE	Integer	Pointers to DEs of associa properties
.	.	.	
.	.	.	
.	.	.	
4+N+M	DE	Integer	



EXAMPLE 1 EXAMPLE 2 EXAMPLE 3

FIG. 3.2-16 EXAMPLES OF ORDINATE DIMENSIONS

POINT DIMENSION

A point dimension consists of a leader, text, and an option circle or hexagon enclosing the text. If a transformation matrix is specified, it transforms the XT-YT plane of the point dimension into model space.

The leader will always contain three segments, and its first and last segments are always horizontal or vertical. If a hexagon encloses the text, it will be described by a composite entity. If a circle or hexagon does not enclose the text, the last segment of the leader will be vertical and it will underline the text.

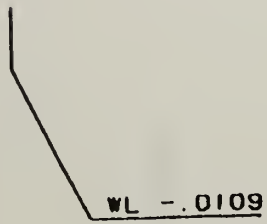
Examples are shown in Fig. 3.2-17.

Directory Data

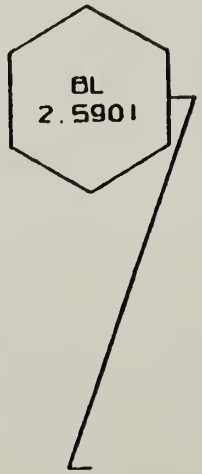
ENTITY NUMBER : 220

Parameter Data

Parameter	Value	Format	Comment
1	DE1	Integer	Pointer to General Note DE
2	DE2	Integer	Pointer to leader DE
3	DE3	Integer	Pointer to Circle, or Composite Entity DE, or 0.
4	N	Integer	Number of associated entities
5	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
4+N	DE	Integer	
5+N	M	Integer	Number of associated properties
6+N	DE	Integer	Pointers to DEs of associated properties
.	.	.	
.	.	.	
.	.	.	
5+N+M	DE	Integer	



EXAMPLE 1



EXAMPLE 2

EXAMPLE 3

FIG. 3.2-17 EXAMPLES OF POINT DIMENSION

RADIUS DIMENSION

A radius dimension consists of a general note, a leader and an arc center point, (x , y). Refer to Fig. 3.2-18 for examples of radius dimensions.

The arc center coordinates are used for positioning the radius dimension line relative to the arc being dimensioned.

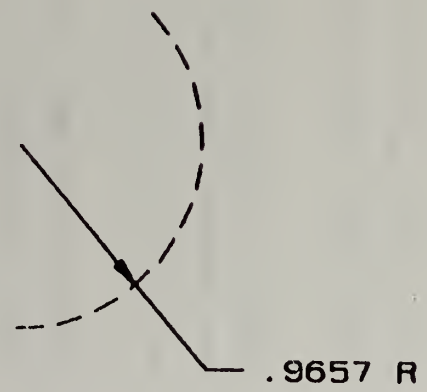
Directory Data

ENTITY NUMBER : 222

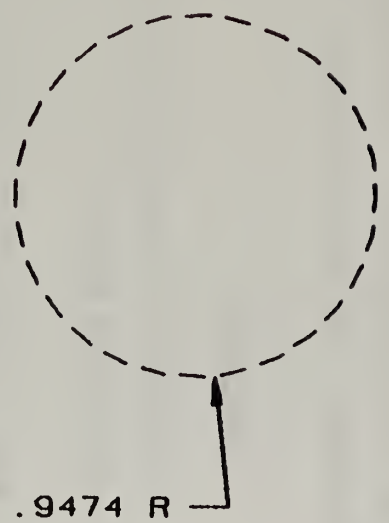
Parameter Data

Parameter	Value	Format	Comment
1	DEN	Integer	Pointer to general note DE
2	APR	Integer	Pointer to leader DE
3	XT	Floating Point	Arc center coordinates
4	YT	Floating Point	
5	N	Integer	Number of associated entities
6	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
5+N	DE	Integer	
6+N	M	Integer	Number of associated properties

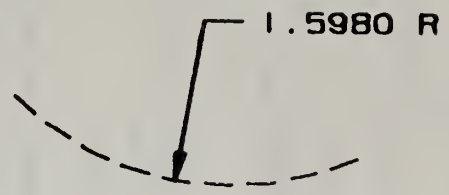
Parameter	Value	Format	Comment
7+N	DE	Integer	Pointers to DEs of associat properties
.	.	.	
.	.	.	
.	.	.	
6+N+M	DE	Integer	



EXAMPLE 1



EXAMPLE 2



EXAMPLE 3

FIG. 3.2-18 EXAMPLES OF RADIUS DIMENSION

SECTION

A section is a copious data entity containing a list of points. The form number describes how the data are to be interpreted.

The point data contains a list of points (Xn, Yn) n=1, 2, . . . , N. (The Z value may be constant)

These points represent line segments of the section display symbol.

The table below describes the use of the form number, and Fig. 3.2-19 shows examples of each possible form number. (Reference: the American Standard Association definition of Symbols for Materials).

Form number	Display action
31	Display all line segments as solid lines. (Iron, Brick, Stone masonry)
32	Do not display every third line. All other lines display as solid. (Steel)
33	Display odd numbered lines as solid. Display even numbered lines dashed. (Bronze, Brass, Copper)
34	Do not display every fifth line. All other lines display as solid. (Plastic, Rubber)
35	Display odd numbered lines as solid. Display even numbered lines with a centerline line font. (Fire brick, Refractory material)
36	Display all lines with a centerline line font. (Marble, Slate, Glass)

Form number	Display action
37	Display all line segments as solid lines. (Lead, Zinc, Magnesium, Sound or heat insulation, Electrical insulation)
38	Display the first half of the lines with a solid line font. Display the last half of the lines with a dashed line font. (Aluminum)

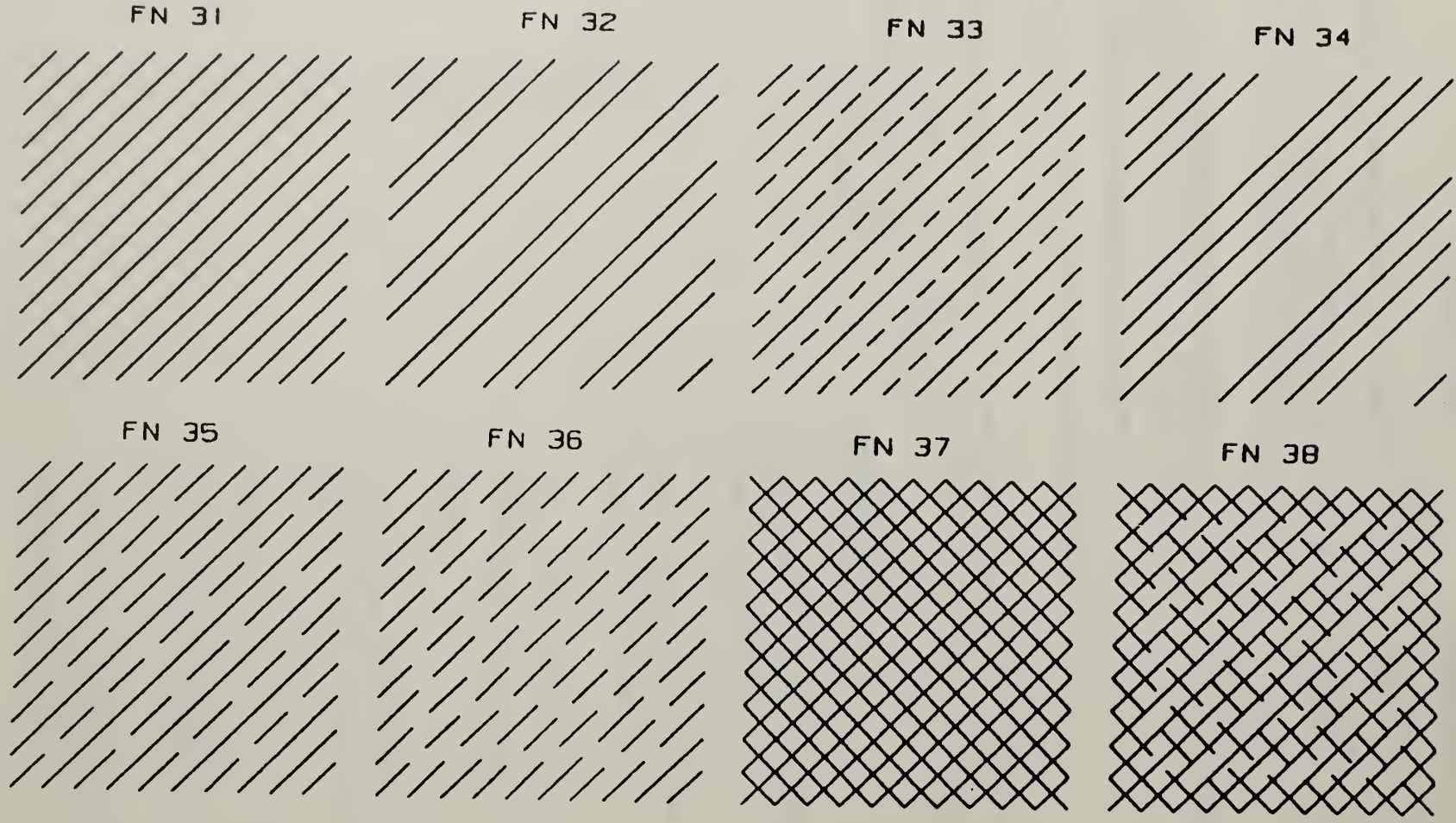


FIG. 3.2-19 EXAMPLES OF SECTION ENTITIES
(FN = FORM NUMBER)

WITNESS LINE

A witness line is a form number 40 of a copious data block that contains one or more straight line segments associated with drafting entities of various types. If a transformation matrix is specified, it transforms the XT-YT plane of the witness line into model space. Refer to Fig. 3.2-20 for examples of witness lines.

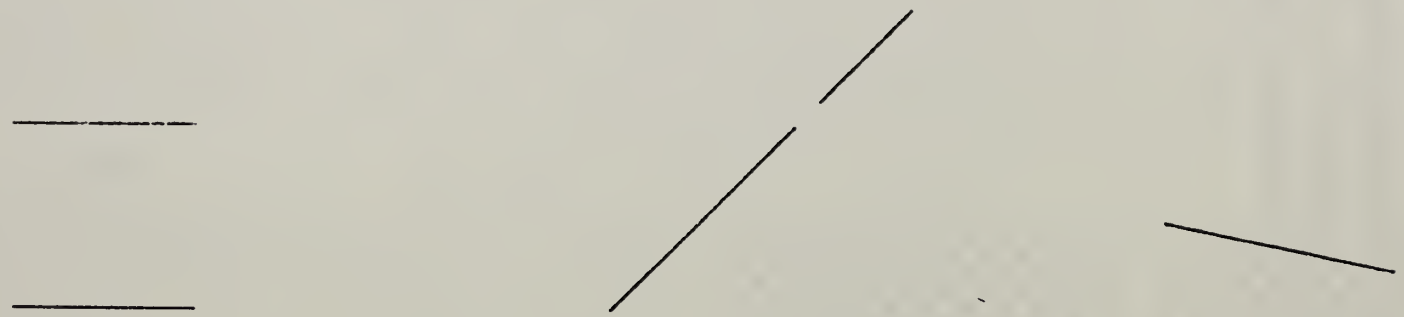


FIG. 3.2-20 EXAMPLES OF WITNESS LINES

3.3 Definition Entities

IGES contains five types of definition entities.

Entity Name	Entity Number
Associativity definition	302
Line font pattern	304
Macro definition	306
Subfigure definition	308
Text font definition	310

ASSOCIATIVITY DEFINITION

The associativity definition entity permits the preprocessor to define an associativity schema. That is, by using the associativity definition, the preprocessor defines the type of relationship. It is important to note that this mechanism specifies the syntax of such a relationship and not the semantics.

The definition schema allows the specification of multiple classes. A class is considered to be a separate list, and the existence of several classes implies an association among the classes as well as between the contents of each class.

For each class, the schema has provision to specify whether or not back pointers are required. A back pointer being required implies that an entity which is a member of this associativity (when it is instanced) has a pointer to the DE of the instance in its parameter section.

The provision in the schema to specify if a class is ordered or not is used to state whether or not the order of appearance of entries in the class is significant.

In the schema "ENTRIES" are the members of the class. However, each entry could be composed of several items. That is, an entry may require more than one item to be complete. For example, if the entries were locations, each entry might have three items to specify X, Y, and Z values.

Finally, in order to help decode instances of the definition, each item is specified as a pointer (to an ENTITY DE) on a data value.

Within the IGES file two forms of associativity are permitted. The first form is a set of IGES-defined associativities which will have form numbers in the range of 1 to 5000. The associativity definition entity defines the second form which are those defined on the IGES file by a preprocessor (Form numbers 5001-9999). The definition appears once in the file for each form of associativity defined, and allows the preprocessor to fill in the definition according to a schema which defines the details of the associativity.

The definition includes the associativity form, the number of class definitions, the number and type of items in each entry, and whether back pointers (from the entity to the associativity) are required. Each set of values (BP, Order, N(i), and Item type) are considered a class. The IGES defined associativity definitions are located in Section 3.5.1. Refer to Section 3.5.1 for a complete example of associativity.

Directory Data

ENTITY NUMBER : 302
ATTRIBUTES REQUIRED : FORM

Parameter Data

Parameter	Value	Format	Comment
1	K	Integer	Number of class definitions
2	BP	Integer	1-back pointers required 2-no back pointers
3	Order	Integer	1-ordered class 2-unordered class
4	N(1)	Integer	Number of items per entry
5	Item type	Integer	1-pointer to a DE 2-value
.			
.			
4+N(1)			

The items in parameters 2 through 4+N(1) are repeated for each of the K classes.

5+N(1)	BP	Specification for second class
6+N(1)	Order	
7+N(1)	N(2)	
.		
.		
.		
7+N(1)+N(2)		
.		
.		
.		
$1+\sum_{i=1}^{K} 3+N(i)$		

LINE FONT PATTERN

This entity is used only to generate Line Fonts. A repeating pattern is specified w on or off segments. A repeating subfigure pattern is specified when M=0. If M parameter 3 indicates the length of the first segment of the subfigure pointed to parameter 2, and parameter 4 is the scale factor to be applied to each instance of subfigure. If M=0 parameters 5 through M+3 will not occur. Fig. 3.3-1 demonstra the use of the Line Font pattern. Figure 3.3-2 demonstrates the use of a subfig (M=0).

Directory Data

ENTITY NUMBER: 304

Parameter	Value	Format	Comments
1	M	Integer	Number of segments in minimum repeating struct of the Line Font (M 16) 0.
2	L1	Floating Point (or Integer)	Length of the first segmen If M=0 this is a pointer t subfigure to be used as repeat pattern.
3	L2	Floating Point	Length of the second s ment. If M=0, L2 is length of the repeat pattern.
4	L3	Floating Point	If M=0, L3 is a scale fac to be applied to subfigure. Otherwise L3 the length of the th segment.
.			
.			
.			
M+1	Lm	Floating Point	Length of the m^{th} segmen
M+2		Character	Up to 4 Hexadecimal nu bers representing whether not a segment is blank or blank. Ex: '5' would indic that segments 1, and 3 w blank.

M = 8

L1 THRU L8 = 1

CHAR = A1

BIT PATTERN = 10100001

RESULTING FONT:

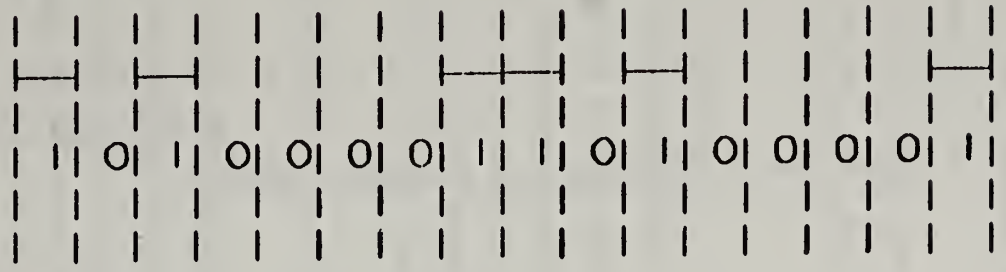


FIG. 3.3-1 CONSTRUCTION OF LINE FONTS

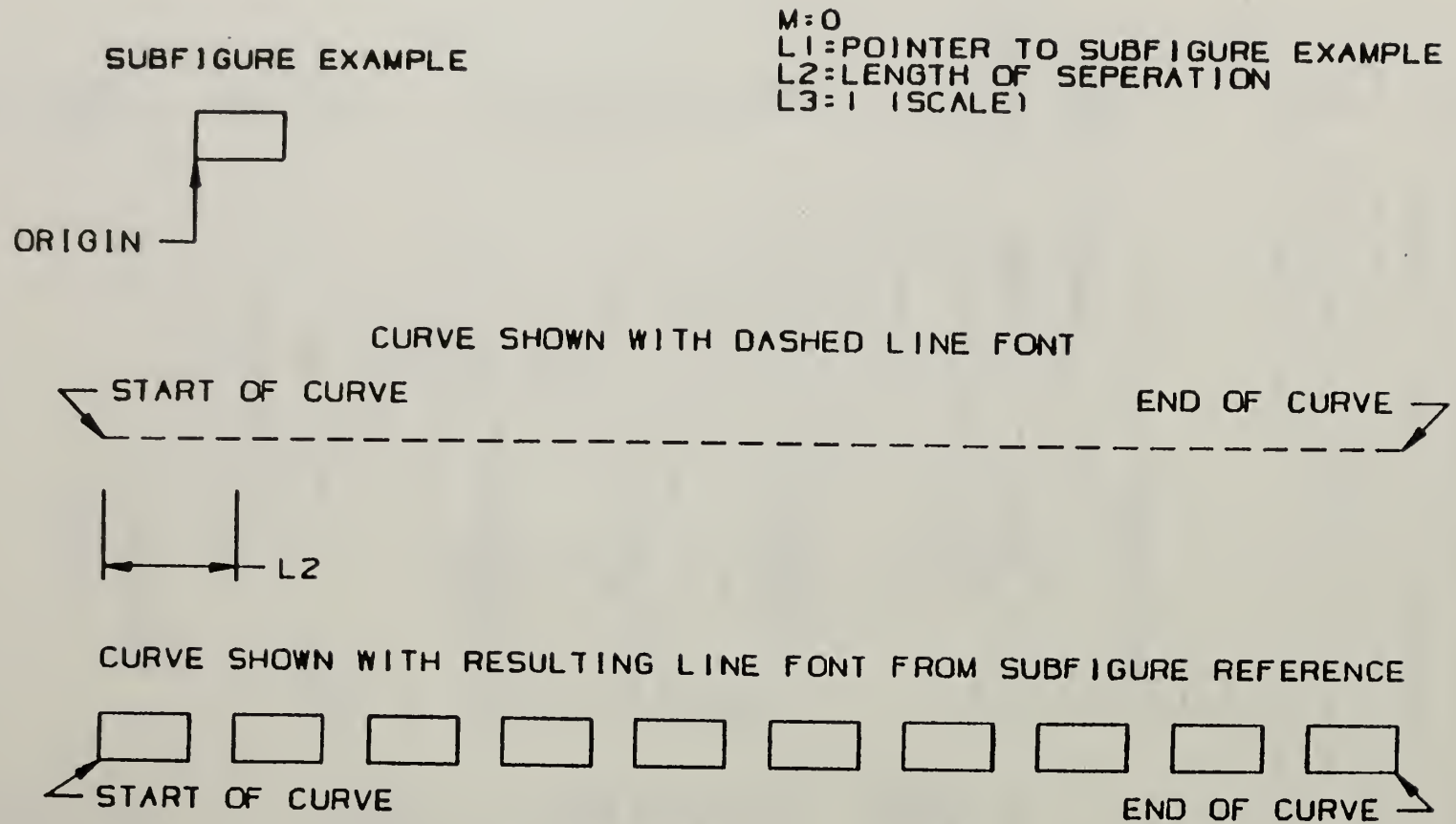


FIG. 3.3-2 USE OF SUBFIGURE WITH LINE FONTS

MACRO CAPABILITY

The IGES specification provides a means for communicating 3D and 2D geometric models and drawings. The specification, however, does not provide a format for every geometric or drafting entity available on all currently used CAD/CAM systems, and is thus a common subject of such entities.

To allow exchange of a large subset of entities - a subset containing some of the entities not defined in the IGES specification, but which can be defined in terms of the IGES basic entities, the marcro capability is provided. This capability allows the use of the specification to be extended beyond the common entity subset, utilizing a formal mechanism which is a part of the specification.
The MACRO capability provides for the definition of a "new" entity in terms of other IGES entities. The "new" entity schema is provided in a macro definition which occurs once for every "new" entity in the file.

A MACRO definition is written using the MACRO entity. The parameter section of the entity contains the MACRO body. In the MACRO body, five types of statements are useable. The statements permissable are the assignment statement (LET), the entity definition statement (SET), and the repeat and continue statements. The repeat statement defines a portion of the body to be repeated and is terminated by a CONTINUE statement. References to other MACRO definitions appear on a MREF statement. The MACRO body is terminated by an ENDM statement.

Each of the statements in a MACRO definition entity is terminated by a semicolon (;).

In order to use a "new" entity defined by the MACRO definition, a MACRO instance is placed in the file. The DE portion of an instance specifies the new entity number in field 1 of the DE pointer and refers to the definition by a pointer in the number field. The parameters to the instance are placed in the parameter section of the instance.

A brief syntax description of the definition statements and examples of instances of a MACRO are presented in the pages which follow. Appendix A contains a complete description of the MACRO capability.

CONSTANTS

REAL	1., 2.5E+01, -3.7842560-01
INTEGER	5, 1121
STRING	3HABC

VARIABLES (6 characters maximum)	FIRST CHARACTER
REAL	(A-H, O-Z)
INTEGER	(I-N)
STRING	$
POINTER	#
ATTRIBUTE	/

FUNCTIONS

SIN, COS, SQRT, ETC (similar to Fortram)

STRING (Real or Integer, Format Field)

STRING (5.0123, F6.4) 6H5.0123

STRING (X, F6.1)

STRING (I, I5)

EXPRESSION (same evaluation rules as Fortran)

OPERATIONS	+,-,*,/,**
OPERANDS	real or integer variables and functions

STATEMENTS

ASSIGNMENT (LET)

ENTITY (SET)

REPEAT

CONTINUE

MREF (Reference to other MACRO)

MACRO

ENDM

ASSIGNMENT STATEMENTS

LET (real or integer variable) = real or integer expression

```
LET X = X+1.0

LET I = 3/2 + 1

LET Dist = SQRT ((x1**2-x2**2)**2 + (y1**2-y2**2))
```

ASSIGNMENT STATEMENT FOR ATTRIBUTE VARIABLE

```
LET/LEV=1

LET/LEV=/HDR
```

NOTE: /HDR is a reserve word meaning reset to the default value for the attribute on the left of the equal sign.

ENTITY STATEMENT

```
SET #ABC=#ARGI

SET #XYZ=110,X1,Y1,Z,X2,Y2,Z,0,0
```

REPEAT STATEMENT

```
REPEAT 5

CONTINUE
```

MACRO STATEMENT

```
MACRO, 610, X1, Y1, X2, Y2, $ABC, #PR

   .
   .
   .
(MACRO BODY)
   .
   .
   .

   ENDM;
```

NOTE:

All statements are terminated by a semicolon in the file.

```
MACRO, 621, X1, Y1, A1, A2, K;

LET Z = 0;

SET #Line1      =    110, X1, Y1, Z, K*(X1+A1), (Y1+A2/2.), Z, 0, 0;

SET #Line2      =    110, K*(X1+A1), (Y1+A2/2.), Z,
                     K*(X1+A1), (Y1-A2/2.), Z, 0, 0;

SET #Line3      =    110, K*(X1+A1), (Y1-A2/2.), Z
                     X1, Y1, Z, 0, 0;

ENDM
```

INSTANCE

```
MACRO, 621, 10., 10., 4., 2.,-1;
```

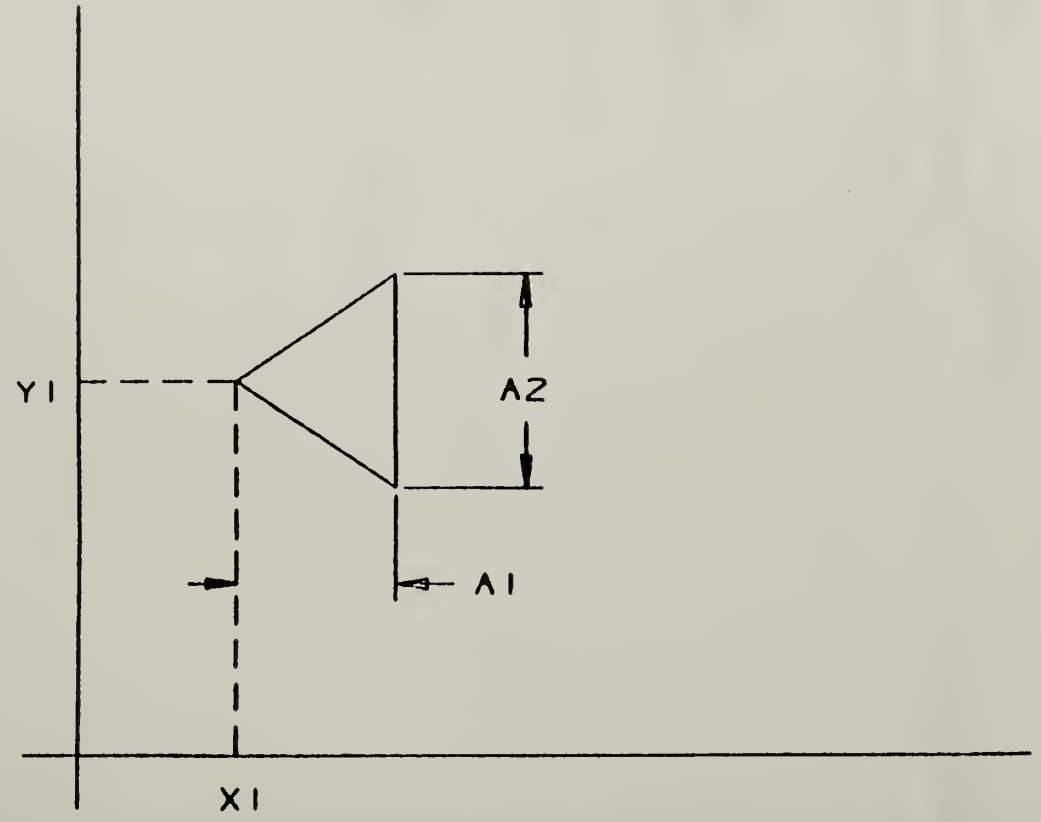


Fig. 3.3-3

```
MACRO,600, X1, Y1, X2, Y2, X3, Y3, N, W, H, Value;
        LET Z = 0.;
        SET #Line = 110, X1, Y1, Z, X2, Y2, Z, 0, 0;
        LET D1 = SQRT ((X1-X2)**2 + (Y1-Y2)**2);
        LET D2 = D1/N;
        LET SLOPE = (Y2-Y1)/(X2-X1);
        LET THETA = ATAN (SLOPE);
        LET DX = D2*COS (theta);
        LET DY = D2*Sin (theta);

        REPEAT (N+1);
        SET #WL = 228, Z, X1, Y1, X3, Y3, 0, 0;
        LET $DIM = STRING (value, F6.3);
        SET #NOTE = 212, 1, 6, (6*w), H, 1, 0., 270., X3, Y3, Z, $DIM, 0, 0;
        LET X1 = X1+DX;
        LET Y1 = Y1+DY;
        LET X3 = X3+DX;
        LET Y3 = Y3+DY;
        LET Value = Value+D2;
CONTINUE;
ENDM;
```

INPUT: COORDINATES OF 3 POINTS AND N, THE NUMBER OF SPACES

+ (X_1, Y_1)

+ (X_2, Y_2)

+ (X_3, Y_3)

RESULTS

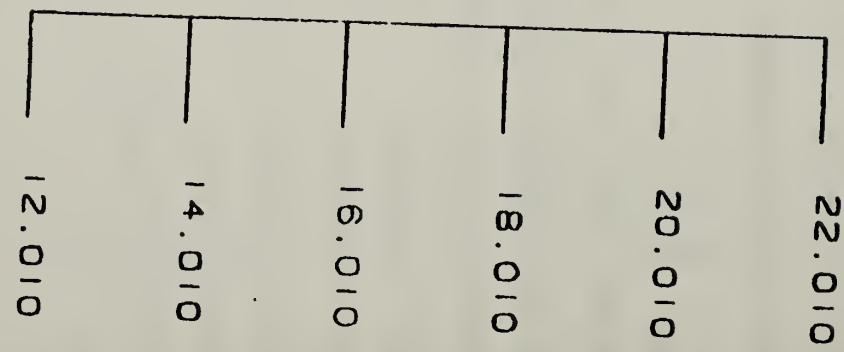


FIG. 3.3-4

SUBFIGURE

The subfigure is designed to support the concept of a subpicture (if one equates drawing creation with graphics picture processing). This entity permits a single definition of a detail to be utilized in multiple instances in the creation of the whole picture. The contents of the subfigure include a set of pointers to any combination of entities and other subfigures. Depth indicates the nesting of the subfigures, and has a range of 0-15.

Directory Data

ENTITY NUMBER : 308

PARAMETER DATA

Parameter	Value	Format	Comment
1	Depth	Integer	Depth of Subfigure (indicating the amount of nesting)
2	Name	Character	Subfigure Name
3	N	Integer	Number of entities in the subfigure
4	DE	Integer	Pointers to the Directory Entries for the Associated Entities
.			
.			
N+3			

TEXT FONT DEFINITION

The text font definition entity is used to define characters and character fonts not provided in the font definitions of IGES. The entity pairs an ASCII value with a subfigure. The subfigure contains the geometric components necessary to draw the character. The first parameter is a pointer to a subfigure which is the default character. Those ASCII codes not mapped to a subfigure, map to the default subfigure, or blank if PD equals zero.

Directory Entry

ENTITY NUMBER : 310

PARAMETER DATA

Parameter	Value	Format	Comment
1	PD	Integer	Pointer to DE of default subfigure
2	N	Integer	Number of ASCII code/Pointer pairs
3	ASCII#	Integer	ASCII character number
4	P1	Integer	Pointer to DE of subfigure
.			
.			
.			
2N+2			

Other Entities

The following section describes other entities available in the specification. These entities are:

Entity Name	Entity Number
Associativity Instance	402
Drawing	404
Property	406
Subfigure Instance	408
View	410

ASSOCIATIVITY INSTANCE

Each time an associativity relation is needed in an IGES file an instance entity is used.

The form number of the associativity instance will identify the meaning of the entity. If the form number is between 1 and 5000, the definition is specified as described in Section 3.5.1, and the version number of the associativity instance will be 1.

If the form number is between 5001 and 9999, an associativity definition will occur in the IGES file and the version number of the instance will be a pointer to the DE of the associativity definition.

Each entity that is a member of an associativity instance can contain a pointer (in the associated entity portion of its parameter data) to the associativity instance. A complete example of associativity is shown in Section 3.5.1.

Directory Entry

ENTITY NUMBER : 402
ATTRIBUTES REQUIRED : FORM, VERSION

PARAMETER DATA

Parameter	Value	Format	Comment
1	N1	Integer	Number of entries for class one
2	N2	Integer	Number of entries for class two
.			
.			
.			
K	NK	Integer	Number of entries for class K

For K classes with (NI,...,NK) Entries
with (I1, . . . , IK) items per entry

Parameter	Value	Type	Comment
K + 1	Class 1	Entry 1	Item 1
			Item 2
			.
			.
			.
			Item I1
		Entry 2	Item 1
			.
			.
			.
			Item I1
		.	
		.	
		.	
		Entry N1	Item 1
			.
			.
			.
			Item I1
	Class 2	Entry 1	Item 1
			.
			.
			.
			Item I2
		Entry 2	Item 1
			.
			.
			.
			Item I2
			.
			.
			.
		Entry N2	Item I2
	.		
	.		
	.		
	Class K	Entry 1	Item 1
X	Class K	Entry Nk	Item IK

Parameter	Value	Type	Comment
X+1	M	Integer	Number of associated entities
X+2	DE	Integer	Pointer to DE of associated entities
.			
.			
.			
X+M+1	DE		
X+M+2	N	Integer	Number of properties
X+M+3	DE	Integer	Pointer to property
.			
.			
.			
X+M+2+N	DE	Integer	Pointer to property

DRAWING ENTITY

The drawing entity defines a collection of drafting entities and views of geometrical entities which, together, constitute a single representation of the part, in the sense that a real drawing constitutes a single representation of a part in standard drafting practice. If desired, multiple drawings can be included in a single IGES file, referring to the same model space. Refer to Fig. 3.4-1 for an example of the use of the Drawing entity.

Directory Data

ENTITY NUMBER : 404

PARAMETER DATA

Parameter	Value	Format	Comment
1	N	Integer	Number of view pointers
2	VPTR1	Integer	Pointer to DE of first view entity
3	XT1	Floating Point	Location in drawing
4	YT1	Floating Point	coordinates of origin of transformed view
5	VPTR2	Integer	Pointer to DE of second view entity
.	.		
.	.		
3N+2	M	Integer	Number of drafting entities(1)
3N+3	DPTR1	Pointer	Pointer to first drafting entity
.	.		in this drawing
.	.		
.	.		

TE=3N+M+2

(1) If only a single drawing entity is present, a zero value implies all drafting entities are to be included.

Parameter	Value	Type	Comment
TE+1	NA	Integer	Number of associ entities
TE+2	DE	Integer	Pointers to DEs associated entities
.	.	.	
.	.	.	
.	.	.	
TE+1+NA	DE	Integer	
TE+2+NA	MA	Integer	Number of associ properties
TE+3+NA	DE	Integer	Pointers to DEs associated properti
.	.	.	
.	.	.	
.	.	.	
TE+2+NA+MA	DE	Integer	

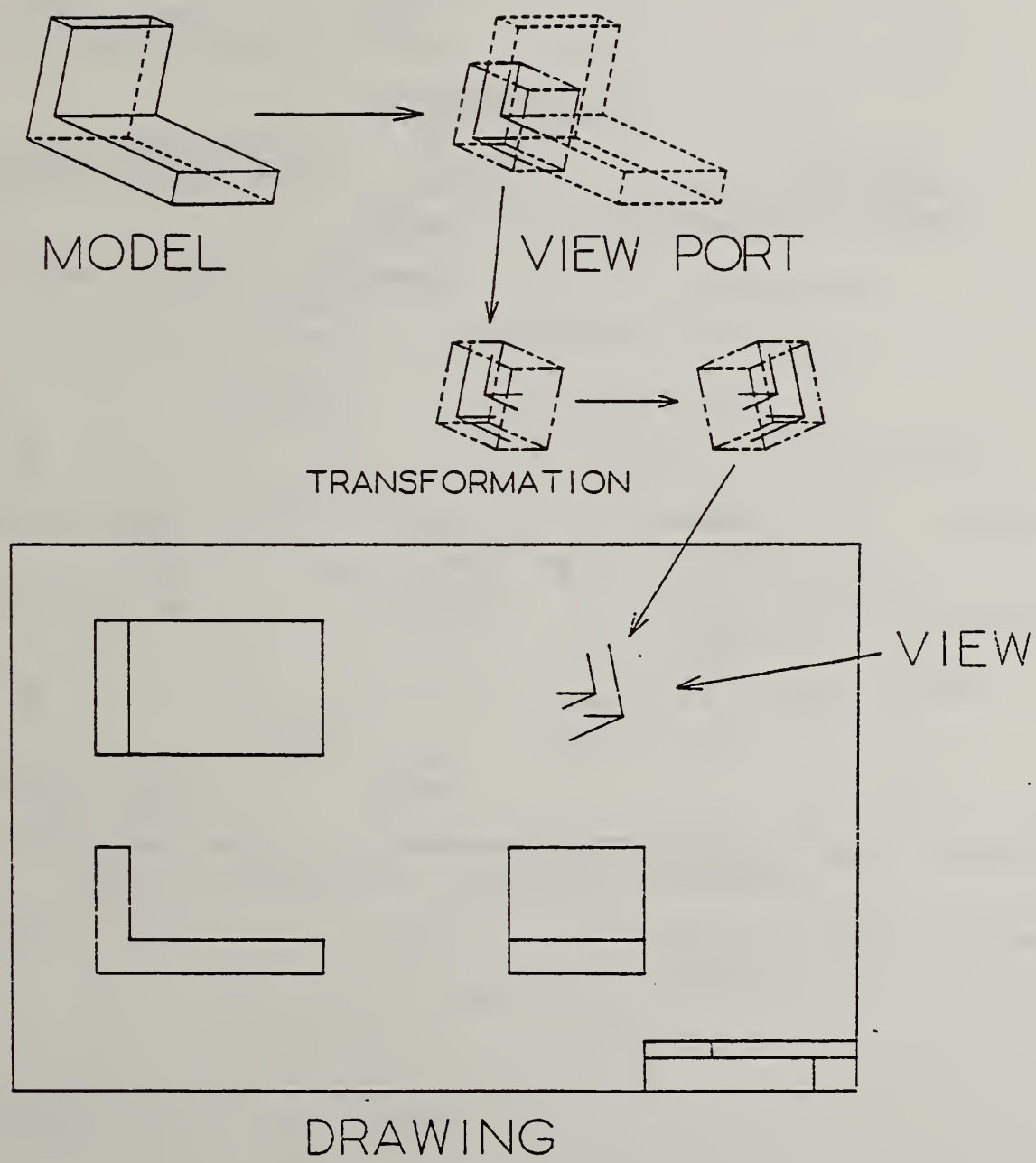


Fig. 3.4-1A

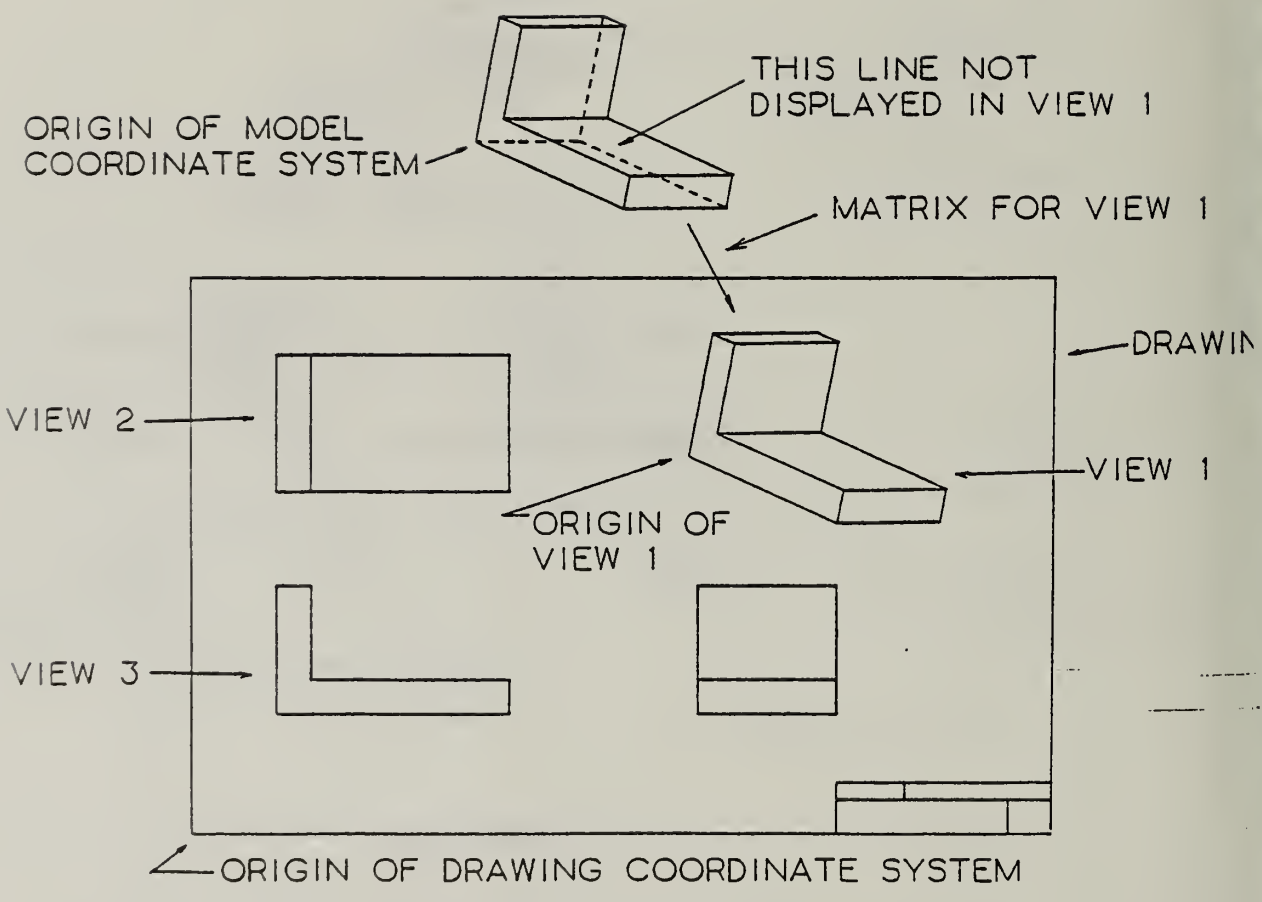

THIS LINE NOT
DISPLAYED IN VIEW 1
ORIGIN OF MODEL
COORDINATE SYSTEM
MATRIX FOR VIEW 1
DRAWIN
VIEW 2
VIEW 1
ORIGIN OF
VIEW 1
VIEW 3
ORIGIN OF DRAWING COORDINATE SYSTEM

Fig. 3.4-1B

PROPERTY

The property entity contains numerical and textual data. It also has a form number to indicate its meaning. Property form number: 1-5000 IGES; 5001-9999 undefined.

Directory Data

ENTITY NUMBER : 406
ATTRIBUTES REQUIRED : FORM

PARAMETER DATA

Parameter	Value	Format	Comment
1	N	Integer	Number of properties
2		Integer	Property Values
.			
.			
.			
1+N			
2+N	NA	Integer	Number of associated entities
3+N	DE	Integer	Pointers to DEs of associated entities
.			
.			
.			
2+N+NA	DE	Integer	
3+N+NA	M	Integer	Number of associated properties
4+N+NA	DE	Integer	Pointers to DEs of associated properties
.			
.			
.			
3+N+NA+M	DE	Integer	

SUBFIGURE INSTANCE

The subfigure instance defines each occurrence of the subfigure.

Before placement by the subfigure instance, the entities in the subfigure are se about the origin of model space. If a matrix reference is specified by the subfi instance, it is then applied to the model space data of the entities in the subfi The model space placement of the subfigure instance is then used to translate subfigure into the model space of the file. Refer to Fig. 3.4-2 for an example o placement of a subfigure.

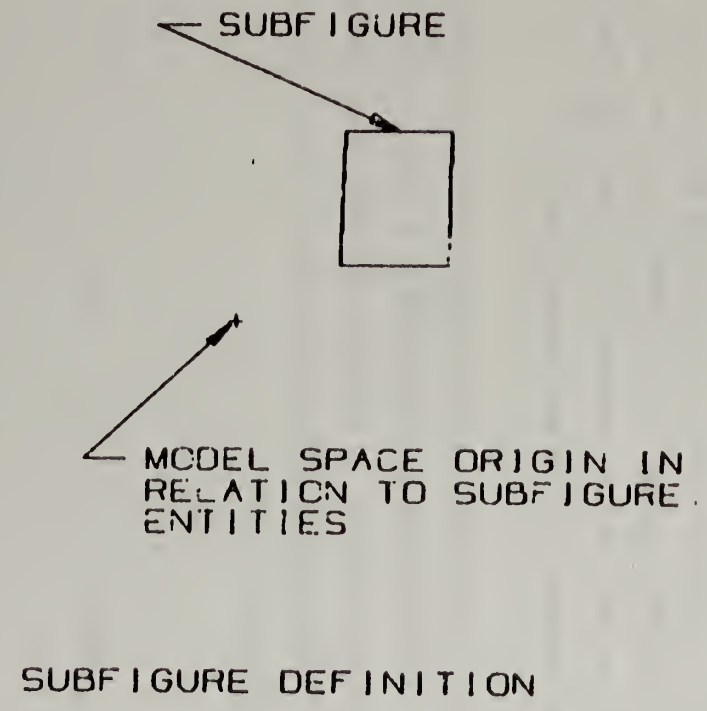
If LEV is specified in the directory data, it overrides that of individual entities.

Directory Data

ENTITY NUMBER : 408

PARAMETER DATA

Parameter	Value	Format	Comment
1	n	Integer	Pointer to subfigure DE
2	x	Floating Point	
3	y	Floating Point	Model space placement subfigure
4	z	Floating Point	
5	s	Floating Point	Scale factor



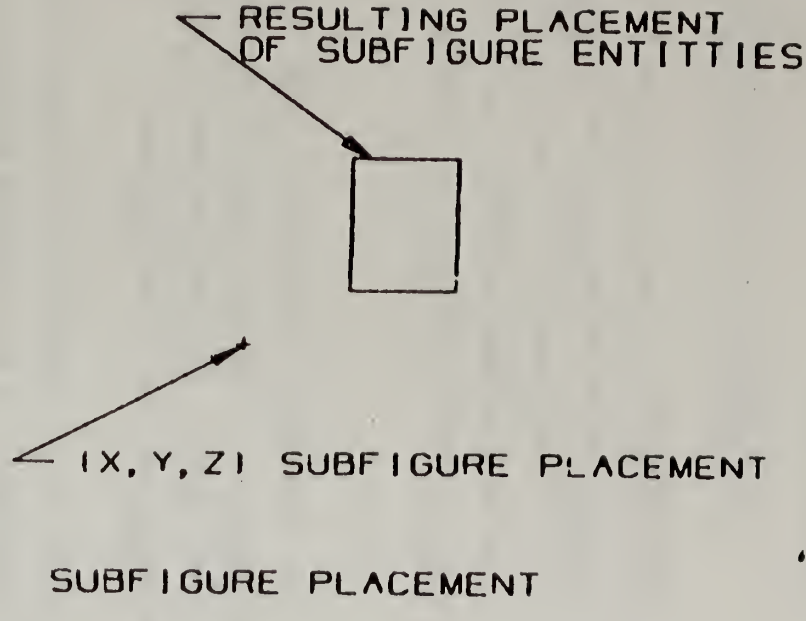


FIG. 3.4-2 SUBFIGURE ORIGIN

VIEW ENTITY

The view entity specifies a specific "look" at the model portrayed by the geometric entities. In its simplest form, the view entity consists of a view number and a pointer to a transformation matrix (in the MTX attribute of the view entity directory entry). This transformation matrix may also be a defining matrix for a set of entities (i.e., the MTX attribute in the entity directory entry specifying the definition plane for the entity).

In more complicated cases, the view entity also defines a scale and optionally a clipping box which control the projection of the view onto a drawing plane. Following transformation by the matrix and clipping, the scale parameter multiplies all model coordinates before they are projected onto the XY drawing plane.

If a clipping box is specified, each entity is clipped at the surface of the clipping box before scaling. This allows only portions of the model to be shown in a particular view. The viewing box is specified by 4 pointers to entities defining the sides and possibly 2 additional entities defining the front and back planes. The entities defining the sides may be either faces or co-planar lines. The front and back are faces. If the front and back are not specified, the clipping box is assumed to extend from plus to minus infinity. Depending on the form of the view entity the parameter list can have 1, 2, 6, or 8 members, exclusive of associativity and property pointers.

The view entity makes possible the selection of display characteristics for each entity (font, weight, pen, etc.). These attributes are specified in a Views Visible (form) associativity associated with a given entity. If a display attribute is not specified, its default is 1. If no Views Visible associativity exists for an entity, the entity is displayed with default attributes in all views.

Directory Data

ENTITY NUMBER: 410
ATTRIBUTES NEEDED: MTX

Parameter Data

Parameter	Value	Format	Comments
1	VNO	Integer	View Number
2	SCALE	Floating Point	Scale Factor
3	FAC1	Integer	Pointer to left side of viewing box
4	FAC2	Integer	Pointer to top of viewing box
5	FAC3	Integer	Pointer to right side of viewing box
6	FAC4	Integer	Pointer to bottom of viewing box
7	BAK	Integer	Pointer to back of viewing box
8	FRO	Integer	Pointer to front of viewing box
9	N	Integer	Number of associated entities
10	DE	Integer	Pointers to DEs of associated entities
.	.	.	
.	.	.	
.	.	.	
9+N	DE	Integer	

Parameter	Value	Format	Comments
10+N	M	Integer	Number associated properties
11+N	DE	Integer	Pointers to of associ properties
.	.	.	
.	.	.	
.	.	.	
10+N+M	DE	Integer	

3.5 Present IGES Definitions

This section contains the specification of predefined definition entities.

3.5.1 Associativity Definitions

As defined in Section 3.3, the Associativity definition entity will only occur for form numbers 5001 through 9999. This following section contains the definition of the defined Associativity definition entities (form numbers 1 through 5000).

Use of Associativity:

To use the associativity, an Associativity definiton must exist. If the form number of the associativity definition is between 1 and 5000, the definition follows in this section and is not required in the IGES file. If the form number is between 5001 and 9999, it will occur in the IGES file. Only one Associativity definition will exist for each form number.

The associativity instance will reference an associativity definition by the form number in its directory entry. If the form number of the associativity instance is between 5001 and 9999, the version number in the directory entry will be a pointer to the DE of the associativity definition in the IGES file. Entities related by an associativity instance may contain pointers (in the associated entities portion of their parameter data) to the associated instance.

Example of Associativity:

This following example is of a group associativity.

If a number of entities were to be grouped together, the predefined associativity definiton form number 1 would be used. If the associativity definition was required in the IGES file, it would appear as shown in Fig. 3.5-1.

The definition of the group associativity indicated there is only one class within the group. This class specifies that only one item per associativity instance entry will occur (N(1)=1), and that this item is a pointer to a DE of an entity (ITEM=1). BP=1 indicates those entities pointed to will also point to the associativity instance. ORDER=1 indicates that when the list of the items occur in the associativity instance, the order is not important.

For each group in the file, an associativity instance similar to the example in Fig. 3.5-2 would occur.

The first parameter of the group associativity instance would indicate the number of entries for the first class (the only class). Since there is only one item per entry and that item is a pointer to an entity, the first parameter indicates the number of entities contained in the group.

Thus, in the example shown in Fig. 3.5-2, the associativity instance points to three entities. These three entities are in the group.

DIRECTORY ENTRY VALUES

ENTITY NUMBER: 302
FORM NUMBER: 1

PARAMETER DATA VALUES

1 - K (NUMBER OF CLASSES): 1
2 - BP (BACK POINTERS): 1
3 - ORDER (ITEMS UNORDERED): 2
4 - N(1) (NUMBER OF ITEMS PER ENTRY): 1
5 - ITEM INDICATOR (POINTERS): 1

FIG. 3.5-1 GROUP ASSOCIATIVITY DEFINITION

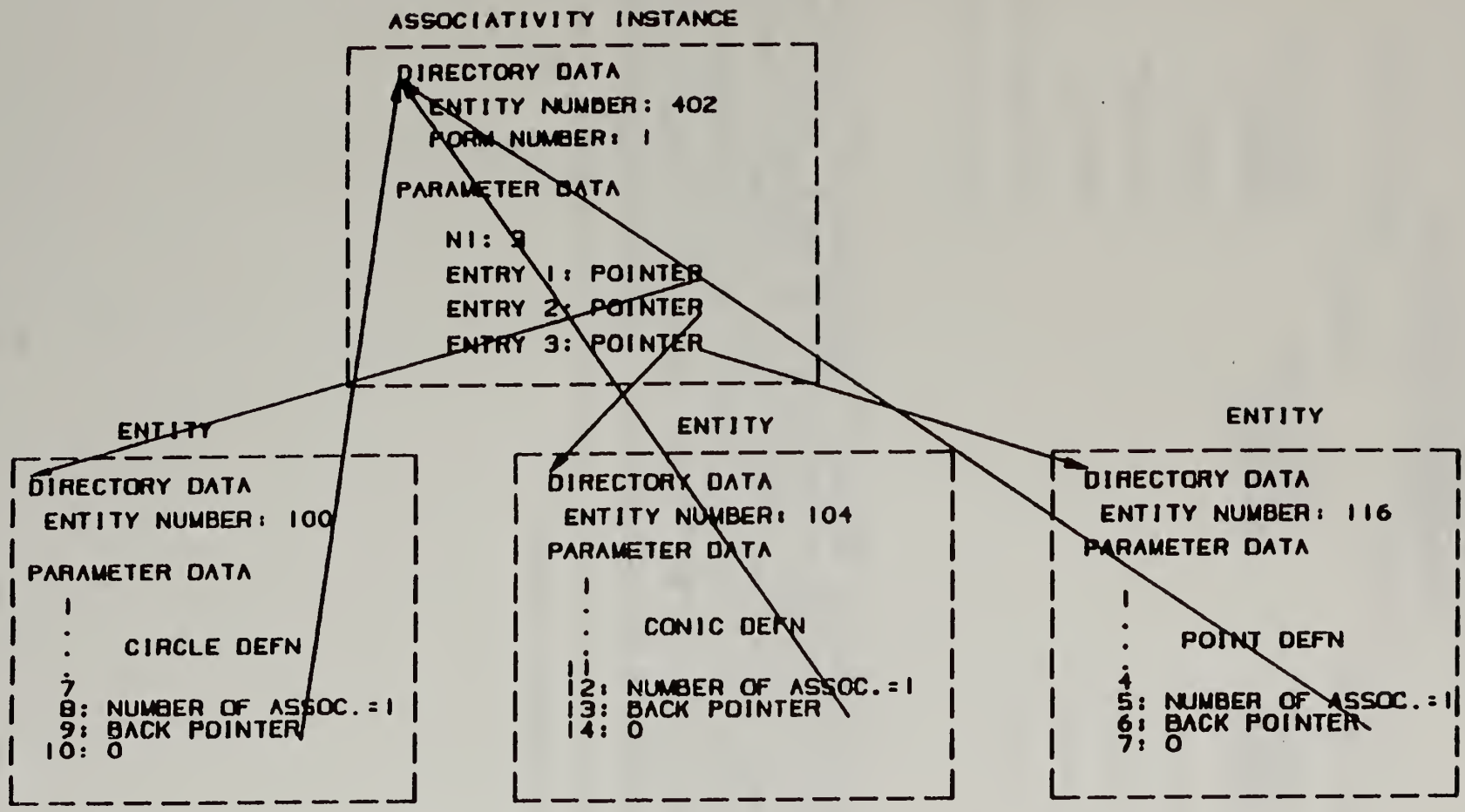


FIG. 3.5-2 ASSOCIATIVITY INSTANCE AND RELATED ENTITIES

IGES DEFINED ASSOCIATIVITY

FORM NUMBER : 1 (Group)

PARAMETER	SET VALUE	MEANING
1	1	One class
2	1	Back pointers
3	2	Unordered
4	1	One item per entry
5	1	The item is a pointer

Since a group associativity definition is defined above, the Association defini entity (Form 1) will not appear in the IGES file. For each group that occurs in an I file, an Associativity instance is created (Refer to Section 3.4). The associati instance will contain in its DE a form number 1. Within the Parameter data portio the associativity instance, the value N1 will contain the number of entities in Group, followed by pointers to each entity in the group. Also, each of those entitie the group will contain a pointer (in the associated entities portion of their parame data) to the associativity instance.

IGES DEFINED ASSOCIATIVITY

FORM NUMBER : 2 (Definition Levels)

PARAMETER	SET VALUE	MEANING
1	1	One class
2	2	No back pointers
3	2	Unordered
4	1	One item per entry
5	2	The item is a value

For each entity in the IGES file that is defined on multiple levels, there will be an occurrence of the associativity instance (Form 2). In the parameter data portion of the associativity instance, the parameter N1 will contain the number of multiple levels, followed by a list of those levels. Each entity that is defined on a set of levels will contain a pointer (in the associated entity portion of its parameter data) to the associativity instance listing on which levels the entity is defined.

IGES DEFINED ASSOCIATIVITY

FORM NUMBER : 3 (Views Visible)

PARAMETER	SET VALUE	MEANING
1	1	One class
2	2	No back pointers
3	2	Unordered
4	1	One item per entry
5	1	Item is a pointer (to view entity)

For each entity in the IGES file that is visible in multiple views, there will be a occurrence of the associativity instance (Form 3). In the parameter data portion o the associativity instance, the parameter N1 will contain the number of views visible followed by a list of pointers to the view entity defining the view.

IGES DEFINED ASSOCIATIVITY

FORM NUMBER : 4 (Views Visible, Pen, Line Weight)

PARAMETER	SET VALUE	MEANING
1	1	One class
2	2	No back pointers
3	2	Unordered
4	5	Five items per entry
5	1	Pointer to view DE
6	2	Line font value
7	1	Pointer to line font DE
8	2	Pen number (value)
9	2	Line weight (value)

For those entities that are visible in multiple views, but must have a different line font, pen number, or line weight in each view, there will be an occurrence of the associativity instance (Form 4). In the parameter data portion of the associativity instance, the parameter N1 will indicate the number of blocks containing the view visible, line font, pen number, and line weight specifications. Each block will contain a pointer to the view entity, a line font value or 0, a pointer to a line font DE if the line font value was 0, a pen number value, and a line weight value. The total number of entries in the associativity instance entity is 5*N1+1.

IGES DEFINED ASSOCIATIVITY

FORM NUMBER : 5 Entity Label Display

PARAMETER	SET VALUE	MEANING
1	1	One class
2	2	No back pointers
3	2	Unordered
4	6	Six items per entry
5	1	Pointer to view DE
6	2	XT of text location
7	2	YT of text location
8	2	ZT of text location
9	1	Pointer to leader DE
10	2	Entity label level number

For those entities that have one or more possible displays for their Entity labels, there will be an occurrence of the associativity instance (Form 5). In the parameter data portion of the associativity instance, the parameter N1 will indicate the number of blocks containing label placement information. Each block will contain a pointer to a view entity which specifies a view of visibility. The remaining information (text location, leader, and level number) apply to the label for that view.

IGES DEFINED ASSOCIATIVITY

FORM NUMBER : 6 (View List)

PARAMETER	SET VALUE	MEANING
1	2	Two classes
(Class One)		
2	1	Back pointers
3	2	Unordered
4	1	One item per entry
5	1	Pointer to View DE
(Class Two)		
6	1	Back pointer
7	2	Unordered
8	1	One item per entry
9	1	Pointer to DE of entry visible in view referred to in parameter 5

This associativity has two classes. The first class has only one entry which is a pointer to the DE of a specific view. The second class is a list of entities (Pointers to their respective DEs) which are visible in the view referenced in class one. Since back pointers are required in both classes, it is assumed that the view points to this associativity instance, as do all entities visible in the view.

3.5.2 Macro Definitions

There are no defintions of Macros in existence at this time.

3.5.3 Property Definitions

There are no definitions of Properties in existence at this time.

APPENDIX A: IGES MACRO CAPABILITY

This appendix describes the syntax of the IGES macro capability. Examples of macro usage appear in Sections 3.4 of the IGES reference manual.

Constants:

Constants may be integer, real or double precision. Integer constants are distinguished by a lack of a decimal point and exponent. Reals are distinguished by the presence of a decimal point, or the presence of an E exponent. The decimal point is optional ONLY when the E exponent is present. Double precision constants are distinguished by the presence of a D exponent, which is mandatory; a decimal point is optional. Any constant may be preceded by a "+" or "-", a "+" is assumed if neither is present. Exponents may also contain a "+" or "-". Constants may not contain a comma, but may contain imbedded blanks. Examples of valid constants follow:

Integer	3	4123	13152	+0	-0
	(+0, 0 and -0 are equivalent)				
Real:	-1	3.14159	6.62E-34	1E10	3.1E+3
Double:	1DO	0DO	3.1415926535897D3	-11562.18D-10	

Examples of invalid constants include:

1,000	(commas not allowed)
E10	(need mantissa -- use 1E10 instead)
3.1-06	(E or D cannot be implied)

The limitations on magnitude and accuracy are inherently machine dependant, and are specified in the global section of an IGES file.

Variables:

Variable names may be from 1 to 6 characters in length. The first character must be one of the characters listed below, this character determines the variable type. It is not possible to override the conventions. The 6 character limitation includes the first character. Upper and lower case letters are recognized as distinct, i.e., X is different from x. Variable names longer than six characters may be used; however, only the first six characters will be significant. Variable names may contain imbedded blanks; these blanks are NOT taken as part of the name, so that "A B" is equivalent to "AB." Except for the first character, as outlined below, all characters must be alphabetic (A-Z) or (a-z), or numeric (0-9).

Variable type	First character
Integer	I-N, i-n
Real	A-H, O-Z a-h, o-z
Double precision	!
String	$
Pointer	#

Examples of valid variable names are:

Integer:	IJK	ICOUNT	K101	NTIMES	max
Real:	XYZ	X1	y2	QrsTu1	
Double:	!h	!xi	!Y2	!12341	
String:	$str	$TITLE	$label		
Pointer:	#line	#note	#REF	#XYZ1	

Some invalid variable names:

$$$$ ($ not permitted after first character)

1X43B (1 may not be first character)
A.BC (. is illegal)

Note that there are no "reserved" words; thus a variable name such as "MACRO," which is identical to a statement keyword (described below), will not confuse the interpreter, although it may confuse the user of such a MACRO. It is suggested that these words be avoided.

Functions:

Functions similar to FORTRAN library functions are provided. The rules for mixed mode, however, have been relaxed, so that it is not necessary to use, for example, SQRT(2.) instead of SQRT(2). Note however that functions do have a specific type of value that they return, i.e., integer, real, or double precision. The functions are described here by the type of value returned. The type of argument usually used is also noted: however, as already mentioned it is not necessary to strictly adhere to those rules. For example, either IDINT(!d) or INT(!d) will work equally well, although the meaning might be a little clearer with IDINT(!d). Functions are only recognized in one case (UPPER).

Functions returning integer values:

IABS(i)

Returns the absolute value of i.

ISIGN(i)

Returns 1 if i is positive, 0 if it is zero, or -1 if it is negative.

IFIX(x) or INT(i)

Returns the integer part of x.

IDINT(!d)

Returns the integer part of !d.

IDINT(!d)

Returns the integer part of !d.

Functions returning real values:

FLOAT(i)

Returns a real (floated) value for i, e.g., FLOAT(2) returns "2."

COS(x)

Returns cosine of angle x; angle in radians.

SIN(x)

Returns sine of angle x; angle in radians.

TAN(x)

Returns tangent of angle x; angle in radians.

ATAN(x)

Returns arctangent of x; angle returned in radians.

EXP(x)

Returns natural anti logarithm of x ("e to the x").

ALOG(x)

Returns natural logarithm of x.

ALOG10(x)

Returns common (base 10) logarithm of x.

ABS(x)

Returns absolute value of x.

SQRT(x)

Returns square root of x.

AINT(x)

Returns integer part of x, just like INT; but returns value in floating point form.

SIGN(x)

Returns 1 if x is greater than 0, 0 if x equals 0, and -1 if x is less than 0.

SNGL(!d)

Returns single (real) value of double precision variable !d. As many significant

digits of !d as possible are given to the returned value.

Functions returning double precision values:

DBLE(x)

Returns "double precision"ed value of x. Note that this is merely a conversion, not an extension. Thus, DBL(.333333333) will return .333333333DO, but not .33333333333333333333333333DO. Thus, DBLE(1./3.) is not necessarily equal to 1DO/3DO.

DCOS(!d)

Returns cosine of angle !d; angle in radians.

DSIN(!d)

Returns sine of angle !d; angle in radians.

DTAN(!d)

Returns tangent of angle !d; angle in radians.

DATAN(!d)

Returns arctangent of !d; value returned in radians.

DEXP(!d)

Returns natural anti-logarithm of !d("e to the !d").

DLOG(!d)

Returns natural logarithm of !d.

DLOG10(!d)

Returns common (base 10) logarithm of !d.

DABS(!d)

Returns absolute value of !d.

DSQRT(!d)

Returns square root of !d.

DSIGN(!d)

Returns 1DO if !d positive, 0DO if zero, -1DO if negative.

Expressions:

Expressions may be formed using the above functions, variables and constants, and the following operators:

addition	"+"
subtraction	"-"
multiplication	"*"
division	"/"
exponentiation	"**"

The operators are evaluated in normal algebraic order, e.g., first exponentiation, then unary negation, then multiplication or division, then addition or subtraction. Within any one hierarchy, operators evaluate left to right. Parentheses may be used to override the normal evaluation order, as in the expression "A*B(B+C)," which is different from "A*B+C." Extra parentheses do not alter the value of the expression; it is a good idea to use them, even if not truly necessary. Examples of expressions include:

```
X + 1.0
-B+SQRT(B**2. - 4*A*C)
I + 1
3.14159/2.
-X
!DEL*(!ALPHA-!BETA)
```

Except for the ** operator, it is never permissable to have two operators next to each other, i.e., not 2*-2, but -2*2 or 2*(-2). Multiplication may not be implied by parentheses, e.g., (A+B)(C+D) is illegal, and AB does not imply A*B, but rather the separate variable "AB."

Mode of expression evaluation:

Unlike FORTRAN, mixed mode (integer mixed with real, etc.) is permitted. Whenever two different types are to be operated upon, the calculation is performed in the "highest" type. Integer is the lowest type, Real is next, and Double precision is the highest. Note however, that this decision is made for each operation, not once for the entire expression. Thus 1/3 + 1.evaluates to 1, because the "1/3" is done first, and it is done in integer mode. Integer mode truncates fractions, and does not round. Therefore, the expression

"2/3+2/3+2/3" has a value of zero.

Statements:

There are seven basic statements that can be used. They are:

LET
SET
REPEAT
CONTINUE
MACRO
MREF
ENDM

These "keywords" are recognized only in upper case, and every statement must begin with one of these keywords. Statements are free format, blanks and tabs are ignored, except within strings. Statements may extend over several card images, or more than one statement may be present on a card. All statements are terminated by a semicolon, which must be present.

LET statement

This is the basic assignment statement, and is equivalent to the LET statement of BASIC. The format of a LET statement is:

LET variable = expression;

The expression and the variable may be integer, real, or double precision; they need not be of the same type. Note that this is an assignment statement, and not an algebraic equality. All of the variables on the right hand side of the expression must have been previously defined; it cannot be assumed that variables will default to zero if they are undefined. Some examples of legal LET statements:

```
LET HYPOT = SQRT(A**2+B**2);
LET X = X + 1;
LET ROOT1 = -B + SQRT(B*B - 4*A*C);
```

```
LETI = i;
LET !XYZ = I * 2;
LET START = 0;
```

String variables:

String variables allow characters to be manipulated. String variables may be used in statements almost anywhere that any other variable type may be used; exceptions are noted below.

String variable may be used in LET statements; note that they may not be mixed with any other type of variable in a LET statement, and also note that operations (i.e., +,-, *, etc.) are not possible with string variables. Two forms of LET statement for string variables are possible:

```
LET $str = 23Hstring of 23 characters;
or
LET $stri = $str2;
```

In the first case, the 23 characters following the H are assigned to the string variable $str. In the second case, the string "$str2" is copied into "stri." Examples of these statements include:

```
LET $title = 3HBox;
LET $label = 6hBottom;
LET $x = $label;
```

Note that if a string variable appears on the right hand side of the statement, it must have been previously defined. Spaces are not ignored within a string constant; they become part of the string. Any ASCII character may be part of a string.

There is one other form for setting up a string, it involves the STRING function. The STRING function may only appear in this form, specifically, it may not appear in SET statement argument lists, or MACRO or MREF statements. However, string constants, such as "6hstring" and variables such as "$x" may appear in SET statements and MACRO

statements.

The form of a STRING function statement is:

```
LET $str = STRING(expression,format);
```

Where "expression" is any normal integer, real or double precision expression, "$str" is a string variable name, and "format" is a format field similar to the format fields useable in a FORTRAN FORMAT statement. The allowable format fields are:

```
Iw
Fw.d
Ew.d
Dw.d
```

The effect of this statement is to convert the numeric value of the expression into characters, i.e., the statement:

```
LET $PI = STRING(3.14159,F7.5);
will result in the same thing as
LET $PI = 7H3.14159;
```

Of course, the usefulness of the STRING function is that expressions can be converted, rather than constants. Thus:

```
LET x = 1;
LET y = 2;
LET $xyz = STRING(x+y+1,F5.0);
will result in the same thing as
LET $xyz = 5H  4.;
```

The rules for the format fields follow the standard FORTRAN convention. "Iw" causes integer conversion, resulting in "w" characters. "Fw.d" cause real conversion, resulting in "w" characters, with "d" characters after the decimal point. "Ew.d" results in real conversion, but using an exponent form, and "Dw.d" is the same as "E," but for double

values. Note that this is one place where mixed mode is not allowed — the type of format field and the type of the result of the expression must be identical.

SET statement

The SET statement establishes directory and paremeter data entries for the specified IGES entity. There are two possible forms for this statement. The first form is:

SET #ptr = entity number, argument list;

"#ptr" is a pointer variable, such as "#XYZ," "entity number" is an IGES entity number, such as "110," and "argument list" is a group of variables which are to be written in the paremeter data section of the entity.

Examples of this type of SET statement:

```
SET #LINE = 110,X1,Y1,Z,X2,Y2,Z,0,0;
SET #ABC = 228,Z,A+B/C,Y1,X2,Y2+1,0,0;
SET #qwe = 264,15Hstrings allowed, X,Y,$this2;
```

The argument list may contain expressions, and may spread over more than one card. At least one argument must be present, i.e., the argument list may not be null. The entity number may not be an expression, it must be an integer constant. The pointer variable will be assigned a value corresponding to the directory sequence number of the directory entry of the entity created.

The format of the argument list written out in the paremeter data section depends on the type of argument in the list. Integer arguments will be written in integer format, reals as reals, and doubles as doubles. Thus, functions such as FLOAT, INT, DBLE, etc., might be of use in order to force an expression that, for example, would normally evaluate to an integer value to be written as a real number.

Note that the STRING function is not allowable in a SET statement -- use a separate LET statement with a string variable instead.

The second form of SET statement has the following syntax:

```
SET #ptri = #ptr2;
```

This causes "#ptri" to be assigned the same value as is currently in "#ptr2." This is analagous to a LET statement, but note that pointer variables may not appear in LET statements. There may be no expressions in this statement, nor may there be constants. The pointer on the right hand side should already be defined.

REPEAT statement

The REPEAT statement causes a group of statements, terminated by a CONTINUE statement, to be repeated a specified number of times. The form of a REPEAT statement is:

```
REPEAT expression;
```

The expression is evaluated, and the resulting value is the number of times the statements will be repeated. The expression may be of integer, real or double type; in the case of real or double expressions, the result is truncated to determine the repeat count. If the repeat count is zero or negative, the group of statements is still executed one time.

Examples of REPEAT statements:

```
REPEAT 3;
REPEAT N+1;
REPEAT 0;          (will still go through one time)
REPEAT X+Y;
```

There is a limit of ten REPEAT statements that may be nested inside one another.

After a REPEAT statement such as REPEAT N; it is legal to alter the value of N, this does not affect the repeat count. Also, note that REPEAT is unlike a FORTRAN "DO," because there is no variable being incremented on every pass.

CONTINUE statement

The CONTINUE statement marks the end of a REPEAT group. The form of a CONTINUE statement is:

```
CONTINUE;
```

When a CONTINUE statement is encountered, the repeat count is decremented by one, and checked to see if it is greater than zero. If it is, the interpreter goes back to the first statement after the most recent REPEAT. If not, then the next statement is processed. The number of REPEAT's and CONTINUE's in a macro should be the same. CONTINUE is not implied by ENDM.

MACRO statement

The MACRO statement is used to signify the start of a macro definition. The format of a MACRO statement is:

```
MACRO, entity number, argument list;
```

Where "entity number" is the entity number of the macro, and "argument list" is a list of parameters that are to be assigned values at execution time. The argument list may not be null. The first statement in every macro definition must be a MACRO statement. Note that the argument list may not contain expressions; only symbolic variable names, of type integer, real, double precision, string, or pointer.

The MACRO statement must be the first statement of the macro; and there may be no other MACRO statements inside a macro. Use the MREF statement to reference other macros; but defining a macro inside of another macro is meaningless.

Examples of a MACRO statement:

```
MACRO,610,X1,Y1,X2,Y2,$ABC,#PR;
```

```
MACRO,600,A,B,C;
MACRO,600,!X,!Y,I;
```

ENDM statement

ENDM signifies the end of a macro. The form of an ENDM statement is:

```
ENDM;
```

All macros must have an ENDM statement as their last statement. ENDM is not implied by the end of the parameter data section.

MREF statement

The MREF statement is used to reference another macro from inside a macro definition. The format of a MREF statement is:

```
MREF,ptr,entity number,argument list;
```

Where "ptr" may be either a pointer variable or an integer expression; the value refers to the DE block of the definition of the macro being referenced. "Entity number" is the entity number of the macro being referenced, and "argument list" is an argument list exactly like that of a SET statement. The effect of the argument list is to replace the symbolic names found in the MACRO definition with the value of the expressions contained in the MREF statement, so that execution of the referenced macro will start with the appropriate values. Note that MREF does not start expansion of the referenced MACRO, rather it creates an entity entry which may later be expanded. It is thus not possible for a MACRO being used in the current macro; nor is it possible for the macro being referenced to have access to any of those values except for those in the argument list. Even then, it is not possible for the occurence of a MREF statement to alter any of those values.

Examples of MREF statements:

```
MREF,#mac1,600,X1,Y1,Z1,X2,Y2,3.1;
MREF,33,621,A,B,3+X/W+1,6*W,3.,0,6Hstring,$x;
```

When a MREF statement is encountered during a MACRO expansion, pertinent information will be printed out to enable the referenced macro to be expanded in another pass. Note that it is not strictly necessary for the values in a MREF statement to be of the same type as the values in the definition MACRO, within certain limitations: Integer, Real, and Double values may be freely mixed, although it might be considered a good idea not to do so. String values may only appear where string variables appear in the definition. Pointers may or may not be mixed with integer type; this is machine dependant and so should be avoided. Pointers may never be mixed with reals or doubles.

Attributes:

Attributes may be set using the LET statement. The format for doing so is:

```
LET /attribute name = expression;
or
LET /attribute name = /HDR;
```

The first form allows an attribute to be set to any constant value, including numeric expressions. Note however that the use of expressions is discouraged. Attributes may also be set to string constants, or variables, but not the result of a STRING function. Some examples of the first type of assignment:

```
LET /LEV = 1;
LET /VIEW = 3;
LET /LABEL = 6HBottom;
LET /LABEL = $X;
```

The second form allows restoring an attribute to its default value. Examples:

```
LET /LEV = /HDR;
LET /LABEL = /HDR;
```

The word "/HDR" is the only non-constant that is allowed on the right hand side of an attribute assignment statement. The effect is to restore the value of the attribute to what it was in the directory entry for the instance, or in some cases to a specified default value. The defaults are dscribed below.

Attribute may not be mixed with any other variable type; nor may they appear anywhere but in the above two forms of LET statements.

The allowable attribute names, and their defaults are given here. A default of /HDR indicates that the attribute defaults to the value in the directory entry of the instance.

/LFP	/HDR
/LEV	/HDR
/VIEW	/HDR
/MTX	/HDR
/CE	0
/BS	/HDR
/SE	subordinate
/ET	/HDR
/LW	/HDR
/PN	/HDR
/FORM	0
/LABEL	(blank)
/SUB	0

APPENDIX B: IGES SPLINE REPRESENTATIONS

Introduction

IGES includes two different types of spline representations:

1. A Parametric Piecewise Cubic Polynomial (for curves)
2. A Grid of General Bicubic Patches (for surfaces).

Most of the spline types used in CAD/CAM systems can be mapped into these representations without change in shape. Spline types supported in IGES include parametric cubics, piecewise linear, Wilson-Fowler, modified Wilson-Fowler, B-splines, cartesian product B-splines, and Coon's patches. Spline types not supported include splines of fourth degree or higher, nonlinear splines, splines under tension, rational cubics, and extended Coon's patches.

Spline Functions

In IGES, spline curves are represented by a number of cubic spline functions, one for each of the X,Y,Z coordinates. Each a cubic spline function S(u) is defined by

1. N: The number of segments,

2. t(1),...,t(N+1): The endpoints and the breakpoints separating the cubic polynomial segments,

3. a(i),b(i),c(i),d(i), i=1,N+1: The coefficients of the polynomials representing the spline in each of the N segments (the N+1st segment is not requried to define the spline, but is included to make the endpoint value and derivative available without evaluating the polynomial),

4. CYTPE: The spline type. (1=linear, 2=quadratic, 3=cubic, 4=Wilson-Fowler, 5=Modified Wilson-Fowler),

5. H: The number of continuous derivatives.

To evaluate the spline at a point "u", first determine the segment containing "u", i.e., the segment "i" such that $t(i) \le u < t(i+1)$, then evaluate the cubic polynomial in that segment, i.e., compute

$$S(u) = A(i) + B(i)*(uu) + C(i)*(uu)**2 + D(i)*(uu)**3$$

where $uu = u - t(i)$.

The polynomial is written in terms of the relative displacement uu (rather than u) so that the values of the spline at the breakpoints can be read directly out of the representation (i.e., $S(t(i)) = A(i)$, $i=1,N$). Computations using the relative displacement also have less floating point roundoff error.

This particular "piecewise polynomial" form is only one of many used to represent the spline segments in CAD/CAM systems. Other representations employed include:

1. End points E1,E2 and end slopes S1,S2: The spline can be evaluated using the "Hermite" basis (see p.59 deBoor).

2. Values at four points: The spline value can be computed from the Lagrange or Newton interpolation formulae (see deBoor).

3. End points and "control" points: There are a number of schemes for computing splines from control points which will not be described here.

With a little algebra (e.g., in deBoor), any of these representations can be converted into any other.

Splines can also be represented as a linear combination of the B-spline basis functions. In CAD/CAM systems, B-splines have been used directly in curve fitting (e.g., the B-spline Bezier polygon (Gordon and Riesenfeld)) and indirectly in various spline calculations (e.g., computing a cubic spline interpolate). For every set of breakpoints $t(1),\ldots,t(N+1)$ and degree of continuity h, a set of B-spline functions $B(1,u),B(2,u),\ldots,B(n*$

can be constructed (see deBoor). Then, for any piecewise polynomial S(u) with these breakpoints and continuity there is a set of B-spline coefficients a(1),... a(n*) such that S(u) can be represented as a linear combination of these B-splines

$$S(u) = a(1)*B(1,u) + a(2)*B(2,u) + \ldots + a(n^*)*B(n^*,u)$$

where $n^* = (N-1)*(3-d)+4$.

B-splines can be computed from piecewise polynomials and vice versa (see p.116 deBoor and subroutine BSPLPP (deBoor)).

Several other types of spline representations (e.g., cardinal bases) have been employed, but they are much less common and do not appear to present a problem for IGES.

Spline Curves

Since curves in CAD/CAM problems are frequently many-valued, spline functions cannot represent such curves adequately. The most common approach to curve fitting is to parameterize the curves, i.e., to represent each curve as two or three spline functions (one for each coordinate)

$$X(u) = Sx(u),$$
$$Y(u) = Sy(u),$$
$$Z(u) = Sz(u),$$

which sketch out the curve as the parameter u varies from t(1) to t(N+1). All of the spline function representations of the previous section can be generalized to parametric curves and the algorithms for converting spline curves from one representation to the other follow easily from multiple applications of the corresponding function conversion algorithms.

Wilson-Fowler Curves: In the early sixties, the Wilson-Fowler spline (a special case of parametric cubics) was developed for curve fitting (see APT). It is still used in many turnkey drafting systems but is generally being phased out in favor of

parametric cubic splines. In the Wilson-Fowler representation, each spline segment is defined in a separate coordinate system whose X-axis begins at one endpoint of the segment and passes through the other. Each spline segment is then defined by a cubic spline function Swf(x) and the coordinates of the two endpoints. These Wilson-Fowler splines can be converted to IGES splines by rotating the parametric spline (u,Swf(u)) back into the current coordinate system; however, most types of IGES splines cannot be converted to Wilson-Fowler splines.

Spline Surfaces

The IGES spline surface is the analog of the IGES spline curve, i.e., it is also pieced together out of other primitive functions. The surface is a grid of parametric bicubic patches defined by:

1. M: The number of grid lines in u,

2. tu(1),...,tu(M+1): The grid lines in u,

3. N: The number of grid lines in v,

4. tv(1),...,tv(N+1): The grid lines in v,

5. Ax(i,j),Bx(i,j),...,Ay(i,j),..., Az(i,j),..., i=1,M; j=1,N: The (M+1)*(N+1) sets of 3*16 coefficients defining the bicubic polynomial for each of the 3 coordinates of the patch. As for the parametric curve, additional patches not strictly required to define the surface are included to make the edge values and derivatives available without explicitly evaluating the polynomial,

6 CTYPE: The spline type. (1=linear, 2=quadratic, 3=cubic, 4=Wilson-Fowler, 5=Modified Wilson-Fowler),

7. PTYPE: The patch type. (1=tensor product, 0=general),

8. H: The number of continuous derivatives.

To evaluate the spline at a point "u,v", first determine the patch containing the point "u,v" in the parameter grid, i.e., the patch "i,j" such that tu(i) < u < tu(i+1) and tv(j) < v < tv(j+1), then evaluate the bicubic polynomial in that patch, i.e., compute

```
X(u,v) = Ax(i,j) *vv**0 *uu**0  + Bx(i,j) *vv**0 *uu**1
           + Cx(i,j) *vv**0 *uu**2  + Dx(i,j) *vv**0 *uu**3

       + Ex(i,j) *vv**1 *uu**0  + Fx(i,j) *vv**1 *uu**1
           + Gx(i,j) *vv**1 *uu**2  + Hx(i,j) *vv**1 *uu**3

       + Kx(i,j) *vv**2 *uu**0  + Lx(i,j) *vv**2 *uu**1
           + Mx(i,j) *vv**2 *uu**2  + Nx(i,j) *vv**2 *uu**3

       + Px(i,j) *vv**3 *uu**0  + Qx(i,j) *vv**3 *uu**1
           + Rx(i,j) *vv**3 *uu**2  + Sx(i,j) *vv**3 *uu**3

Y(u,v) = Ay(i,j) . . .

Z(u,v) = Az(i,j) . . .
```

The patches in the IGES spline surface are equivalent to the bicubic surface patch (or the Coon's patch, see p. 170 (Rogers and Adams) for the conversion details). The parameters of the Coon's patch are given as the corner points, corner slopes, and twist vectors (similar in spirit to the point/slope representation for curves).

The Cartesian product B-splines and polynomials are special cases of the IGES spline surface (see p. 176 Rogers and Adams). Each patch on a cartesian product surface is the product of two polynomials, e.g., for the patch coefficients a,b,c,d,e,f,g,

```
X(u,v) =  (a + b*u + c*u**2 + d*u**3)
        * (e + f*v + g*v**2 + h*v**3).
```

A Cartesian product patch can be converted to an IGES patch by computing the tensor product of the coefficients of the u and v polynomials. If an IGES spline surface patch is a Cartesian product, then the coefficients of the u and v polynomials defining it can be extracted from the first row and column of the IGES patch coefficients.

Cartesian product B-splines can also be converted to the IGES form. Fi the B-spline coefficients are converted to Cartesian product patch coefficients, t those coefficients are converted to IGES surface coefficients (the process is descri in Chapter XVII (deBoor) and subroutine BSPP2D (deBoor)).

The Cartesian product IGES form can be converted to Cartesian prod B-splines, but this conversion process has not been studied so thoroughly (we have reference to an existing FORTRAN code). As in the inverse conversion, Cartesian prod polynomial coefficients would be used as an intermediate representation. First, IGES surface is converted to polynomials, then the polynomials are converted to B-spli coefficients by a tensor-product generalization of the 1-D conversion process (see 116 deBoor).

Summary

Most of the conversions are relatively straightforward. The basic algebra covered in standard reference works such as (deBoor), (Coons), and (Rogers and Adam FORTRAN codes are available for many of the more complicated conversion operati (e.g., the B-Spline Code distributed with deBoor). However, because the IGES splin more general than splines found in many CAD/CAM systems (e.g., the APT Wilson-Fow spline). Shape-preserving transformations out of the IGES spline format may not possible. Difficulties encountered include restrictions such as uniform breakpoint spac and smooth second derivatives. In these cases, the conversion must be accomplished by interpolation or smoothing process.

References

1. APT Computer System Manual. Volume 2 - Subroutine Library. IIT Research Institute, 1968.

2. C. deBoor. A Practical Guide to Splines. Springer-Verlag, 1978.

3. S.A. Coones. "Surfaces for Computer Aided Design of Space Forms." MIT Project MAC TR-41, June 1967.

4. W.J. Gordon and R.F. Riesenfeld. "B-Spline Curves and Surfaces." in R.E. Barnhill and R.F. Riesenfeld, ed. Computer Aided Geometric Design. Academic Press, 1974.

5. D.F. Rogers and J.A. Adams. Mathematical Elements for Computer Graphics. McGraw-Hill, 1976.

APPENDIX C: IGES—ELECTRICAL EXAMPLE

It is the purpose of IGES to transfer information from processor to processor in a computer-aided design and manufacturing system. As such, IGES must be able to completely represent a design at any point in its development. To make this point more clear, let us examine a simple electrical circuit shown in Figure 1. We will consider the signal string running from Pin R1 to Pin 6 of device D1.

At the earliest stage in its design, the signal string is represented only in its logical sense. Figure 2 is a model of an associativity representing the logical signal string. Class 1 of the signal string associativity shows its logical structure. In Figure 2, this is the branch in the middle marked R1 and D1-6. The signal name itself, SR1, is represented in the associativity by means of a property.

The logical structure of the string is sufficient, for example, to represent the information necessary for simulating the circuit. That is, if all signal strings were represented as in class 1 of Figure 2, a logic simulator could be run which would simulate the circuit and allow the engineer to verify that the circuit did, in fact, accomplish the design goals. Similar information is necessary for generating test patterns to be applied to the completed circuit to verify that the components and wiring in the circuit are, in fact, correct and that the circuit functions as designed.

At the next stage in the design, our signal string and others like it are represented as a schematic diagram. It is a simple matter to extend the associativity in Figure 2 to pick up the details of the schematic diagram. In the case of SR1, this is represented by class 2 on the left-hand side marked "schematic goemetry."

Each of the pointers represented in Figure 2 as blanks would point to an element in the signal string, i.e., a specific line in the schematic. The complete design, up to this point, is represented by the schematic as shown in Figure 1 plus an associativity, such as the one shown in Figure 2 for SR1, representing each of the signal strings in the schematic.

Up to this point, we have assumed that the information necessary for constructing the logical signal string existed before the schematic diagram was drawn. Our argument

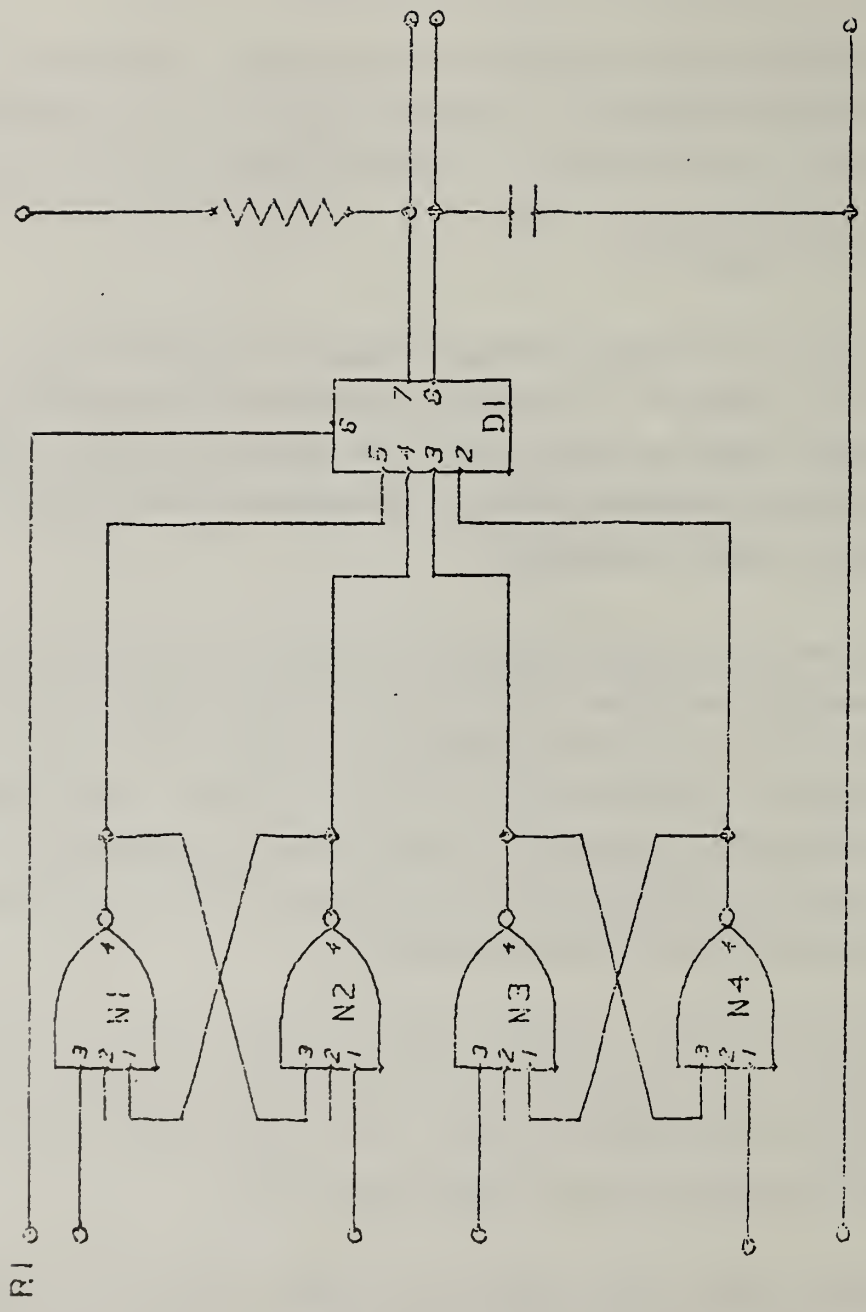
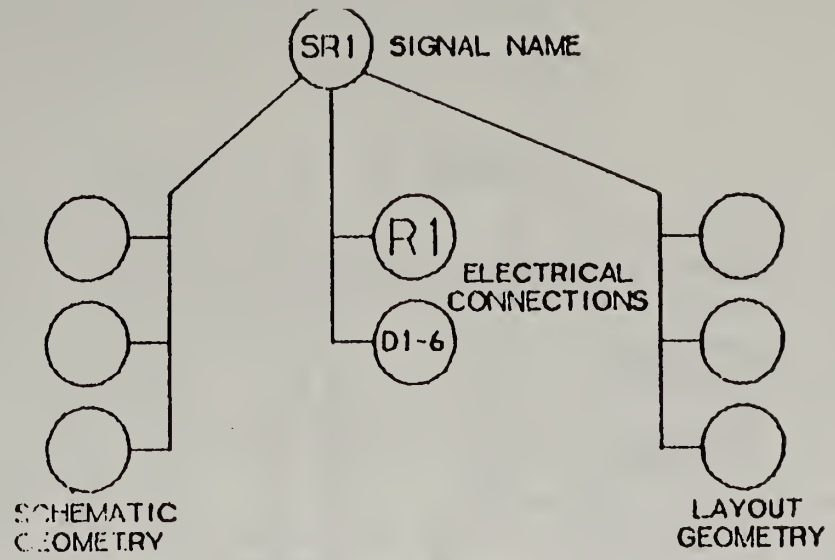
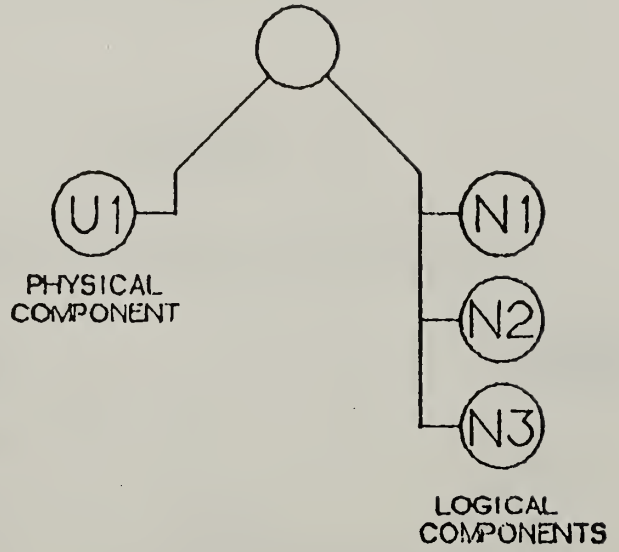


FIG 1



SIGNAL ASSOCIATIVITY

FIGURE 2



PARTITIONING ASSOCIATIVITY

FIGURE 3

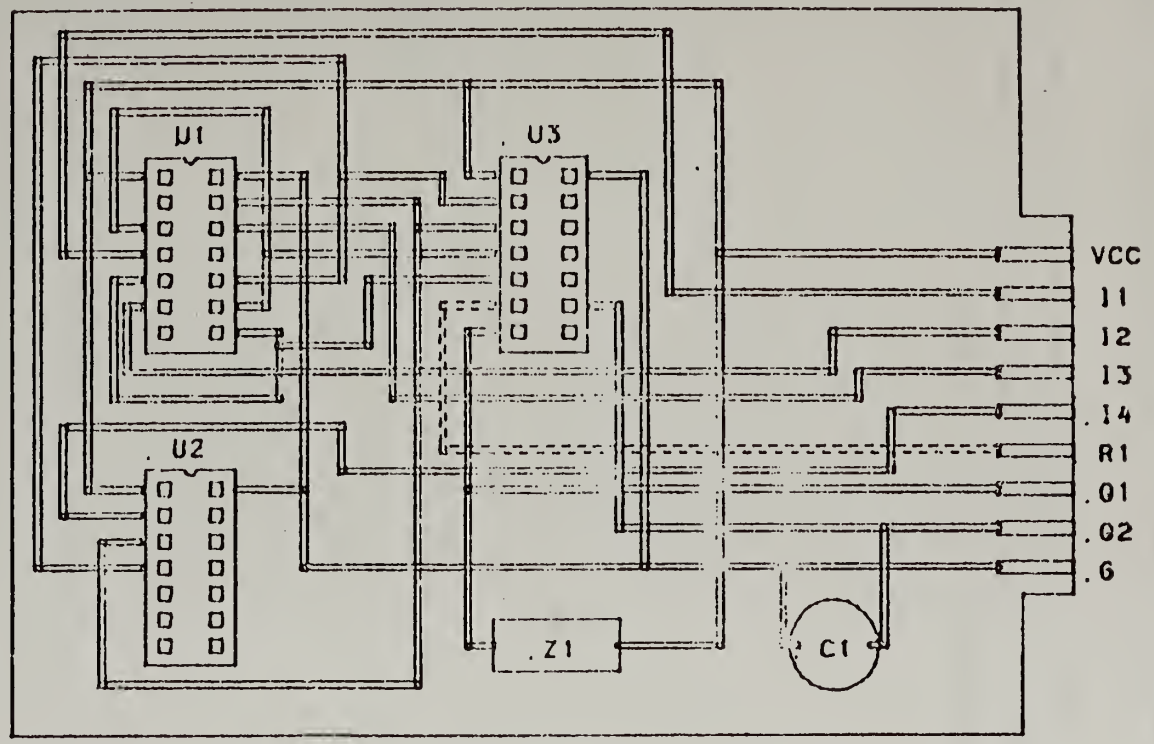


FIGURE 4

APPENDIX D: MECHANICAL PART FILE EXAMPLE

This appendix contains a sample mechanical part encoded in the IGES format. The sample part which is shown in Figure D-1, is two-dimensional and is comprised of lines, circles, a linear dimension, a radius dimension and an angular dimension. The drafting entities, by definition, are made up of witness lines, general notes and leader entities.

The encoded IGES file is shown in Table D-1. On line P0000020 of the table, the degree symbol is shown as a part of the angular dimension entity. This symbol is represented by the octal constant 37 as dictated by the font code of zero, see figure 3.2-11.

TABLE D-1 IGES ENCODED FILE

```
   SAMPLE PART                                                                  S0000
,,11H112C87901.5,11HIGES SAMPLE,6HME1.00,1H1,16,8,24,8,56,11H112C87901.5G0000
,1,1,2H1N,0,.01,13H012880,093243,.1345,800,7HD.MOORE,4HBCAC                     G0000
     12400000001       1                                          000100  D0000
     124                         1                          MTX         1D0000
     11000000002       1         1       10          0          000000  D0000
     110       0       1         1                            L         1D0000
     11000000003       1         1       10          0          000000  D0000
     110       0       1         1                            L         2D0000
     11000000004       1         1       10          0          000000  D0000
     110       0       1         1                            L         3D0000
     11000000005       1         1       10          0          000000  D0000
     110       0       1         1                            L         4D0000
     11000000006       1         1       10          0          000000  D0000
     110       0       1         1                            L         5D0000
     11000000007       1         1       10          0          000000  D0000
     110       0       1         1                            L         6D0000
     11000000008       1         1       10          0          000000  D0000
     110       0       1         1                            L         7D0000
     11000000009       1         1       10          0          000000  D0000
     110       0       1         1                            L         8D0000
     11000000010       1         1       10          0          000000  D0000
     110       0       1         1                            L         9D0000
     11000000011       1         1       10          0          000000  D0000
     110       0       1         1                            L        10D0000
     10000000012       1         1       10          0          000000  D0000
     110       0       1         1                            C         1D0000
     11000000013       1         1       10          0          000000  D0000
     110       0       1         1                            L        11D0000
     21200000014       1         1       1200000055  0          000101  D0000
     212       0       1         1                                    D0000
     21400000015       1         1       1200000055  0          000101  D0000
     214       0       1         1        1                           D0000
     21400000016       1         1       1200000055  0          000101  D0000
     214       0       1         1        1                           D0000
     10600000017       1         1       1200000055  0          000101  D0000
     106       0       1         2       40                           D0000
     21600000019       1         1       1200000055  0          000001  D0000
     216       0       1         1                                    D0000
     21200000020       1         1       1200000055  0          000101  D0000
     212       0       1         1                                    D0000
     21400000021       1         1       1200000055  0          000101  D0000
     214       0       1         1        1                           D0000
     21400000022       1         1       1200000055  0          000101  D0000
     214       0       1         1        1                           D0000
```

TABLE D-1 IGES ENCODED FILE (continued)

```
     10600000023       1       1       1200000055       0       000101  D0000043
     106       0       1       2       40                               D0000044
     20200000025       1       1       1200000055       0       000001  D0000045
     202       0       1       1                                        D0000046
     21200000026       1       1       1200000055       0       000101  D0000047
     212       0       1       1                                        D0000048
     21400000027       1       1       1200000055       0       000101  D0000049
     214       0       1       1        1                               D0000050
     22200000028       1       1       1200000055       0       000001  D0000051
     222       0       1       1                                        D0000052
     10000000029       1       1       10               0       000000  D0000053
     100       0       1       1                                C       2D0000054
     41000000030       1                          00000001       000001  D0000055
     410                       1                                        D0000056
124,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0;                     00000001P0000001
110,2.0175,1.4795,0.0,4.4385,1.4795,0.0;                                 00000003P0000002
110,4.4385,1.4795,0.0,4.4385,2.7535,0.0;                                 00000005P0000003
110,4.7785,3.0935,0.0,5.9180,3.0935,0.0;                                 00000007P0000004
110,5.918,3.0935,0.0,5.918,3.9005,0.0;                                   00000009P0000005
110,5.2455,4.4385,0.0,2.0175,4.4385,0.0;                                 00000011P0000006
110,2.0175,4.4385,0.0,2.0175,1.4795,0.0;                                 00000013P0000007
110,2.421,1.883,0.0,3.228,1.883,0.0;                                     00000015P0000008
110,3.228,1.883,0.0,3.228,3.228,0.0;                                     00000017P0000009
110,3.228,3.228,0.0,2.421,3.228,0.0;                                     00000019P0000010
110,2.421,3.228,0.0,2.421,1.883,0.0;                                     00000021P0000011
100,0.0,4.7785,2.7535,4.7785,3.0935,4.4385,2.7535;                       00000023P0000012
110,5.918,3.9005,0.0,5.2455,4.4385,0.0;                                  00000025P0000013
212,1,6,.48,.1,0,0.0,0.0,0,0,.807,2.8245,0.0;6H2.9590;                   00000027P0000014
214,1,.1,.08,0.0,1.157,4.4385,1.157,3.0245;                              00000029P0000015
214,1,.1,.08,0.0,1.157,1.4795,1.157,2.7245;                              00000031P0000016
106,1,4,0.0,2.0175,4.4385,1.07888,4.4385,2.0175,1.4795,1.07888,          00000033P0000017
1.4795;                                                                  00000033P0000018
216,00000027,00000029,00000031,00000033;                                 00000035P0000019
212,1,8,.64,.1,0,0.0,0.0,0,0,6.187,4.035,0.0,8H38.6597°;                 00000037P0000020
214,1,.1,.08,0.0,6.38174,3.5295,6.17118,3.315;                           00000039P0000021
214,1,.1,.08,0.0,6.70059,4.4385,6.66967,4.7369;                          00000041P0000022
106,1,4,0.0,5.9485,3.8761,6.44274,3.48071,5.28456,4.4385,6.7787200000043P0000023
,4.4385;                                                                 00000043P0000024
202,00000037,00000043,5.2455,4.4385,1.45509,00000039,00000041;           00000045P0000025
212,1,7,.56,.1,0,0.0,0.0,0,0,5.142,2.64,0.0,7H.3400 R;                   00000047P0000026
214,2,.1,.08,0.0,4.53809,2.99392,4.842,2.69,5.042,2.69;                  00000049P0000027
222,00000047,00000049,4.7785,2.7535;                                     00000051P0000028
100,0.0,4.035,3.766,4.4385,3.766,4.4385,3.766;                           00000053P0000029
410,1;                                                                   00000055P0000030
S0000001G0000002D0000056P00000030
```

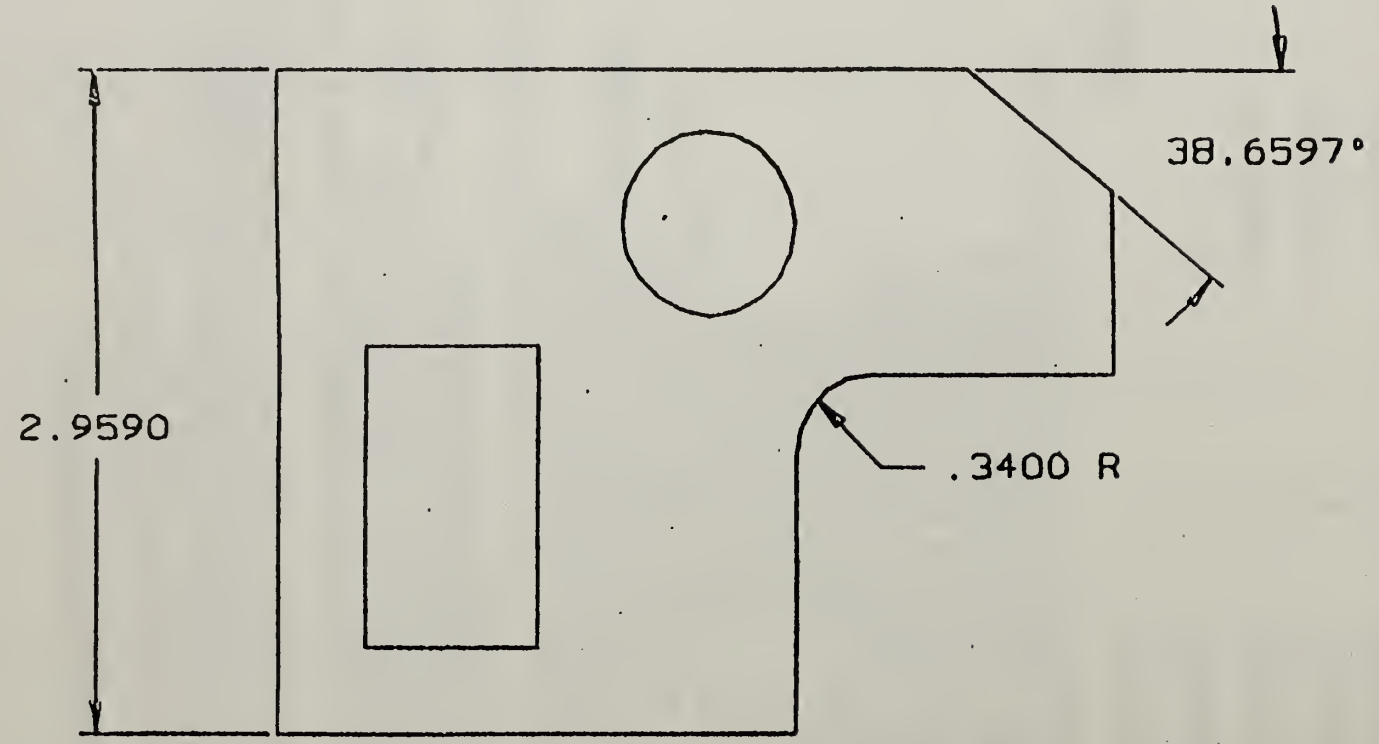


FIG. D-1 SAMPLE PART

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBSIR 80-1978 (R)	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Initial Graphics Exchange Specification (IGES)

5. Publication Date

6. Performing Organization Code

7. AUTHOR(S)

Roger N. Nagel, Walt W. Braithwaite, and Philip R. Kennicott

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, DC 20234

10. Project/Task/Work Unit No.

11. Contract/Grant No.
FY1457-80-00012

12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

Air Force ICAM Program, (Integrated Computer Aided Manufacturing), Wright-Patterson AFB, OH 45433

13. Type of Report & Period Covered

Oct. 1979 - Jan. 1980

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The Initial Graphics Exchange Specification (IGES) contained in this report was developed as a first step in facilitating the communication of data between CAD/CAM systems. The specification contains a defined format for an exchange file written in ASCII characters on 80 column card images. The format consists of entity definitions for geometry, drafting and structural information. In addition, definition entities are provided as a means of expanding the utility of the IGES.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Computer Aided Design (CAD); Computer Aided Manufacturing (CAM); design drawing, exchange format; geometry; graphics; part model

18. AVAILABILITY ~~Unlimited~~

☒ For Official Distribution. Do Not Release to NTIS

☐ Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402, SD Stock No. SN003-003-

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PAGES
UNCLASSIFIED	
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	